

Oracle® Revenue Management and Billing

Version 2.4.0.1.0

Transaction Feed Management (TFM) - Batch Execution Guide

Revision 6.0

E69830-01

March, 2016

Oracle Revenue Management and Billing Transaction Feed Management (TFM) - Batch Execution Guide E69830-01

Copyright Notice

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Trademark Notice

Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure, and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or de-compilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Oracle programs, including any operating system, integrated software, any programs installed on the hardware and/or documentation delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware and/or documentation shall be subject to license terms and restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products, or services.

Preface

About This Document

This document will help you to understand the sequence in which the batches should be executed while performing various tasks in TFM. It also helps you to improve the batch performance.

Intended Audience

This document is intended for the following audience:

- End-users
- Implementation Team
- Consulting Team
- Development Team

Organization of the Document

The information in this document is organized into the following chapters:

Section No.	Section Name	Description
Section 1	Introduction	Provides an overview of the transaction feed management process.
Section 2	TFM Batch Execution Sequence	Explains the sequence in which the batches must be executed while performing various tasks in TFM.
Section 3	TFM Batches	Provides detailed information about all TFM batches.
Section 4	Recommended Parameter Values	Recommends parameter values for each batch.

Related Documents

You can refer to the following documents for more information:

Document	Description
<i>Oracle Revenue Management and Billing Banking User Guide</i>	Lists and describes various banking features in Oracle Revenue Management and Billing. It also describes all screens related to these features and explains how to perform various tasks in the application.

Contents

1. Introduction	1
2. TFM Batch Execution Sequence	2
3. TFM Batches	4
3.1 Rollback (C1-TXNRB)	4
3.2 Flush All Caches (F1-Flush)	6
3.3 Refresh Pricing (C1-TXNRP)	6
3.4 Header Validation (C1-TXNHV)	7
3.5 Transaction Validation and Initial Product Determination (C1-TXNIP)	9
3.6 Product Pricing Verification (C1-TXNVP)	14
3.7 Update Status (C1-TXNEX)	15
3.8 Service Quantity Calculation (C1-TXNSQ)	19
3.9 Mark Completion (C1-TXNCM)	22
3.10 Clean Up (C1-TXNCU)	24
3.11 Disaggregation Request Creation (C1-DISTG)	29
3.12 Pending Bill Segments Deletion (C1-BSEGD)	30
3.13 Pending Bill Deletion (C1-PNBD)	32
3.14 Identify Affected Transactions (C1-IAENT)	33
3.15 Process Non Aggregated Transactions (C1-PDTXN)	35
3.16 Update Disaggregation Request Status (C1-DARSU)	38
3.17 Pending Bill Deletion (C1-DELBL)	39
3.18 Cancellation (C1-TXCNC)	40
4. Recommended Parameter Values	42

1. Introduction

Oracle Revenue Management and Billing provides you with a facility to upload banking transactions received from various product processors or banking applications for billing. Once the transaction data is uploaded in the system, you need to:

- Validate Header Details
- Validate Transaction Details and Determine Initial Product
- Verify Product Pricing
- Rate Transactions Before Billing (if required)
- Create and Update Billable Charge with the SQL values
- Clean-up Unwanted Data

The system provides the flexibility to rate the transactions either prior to billing or during billing. Based on the business requirements, you can configure the system such that transactions mapped to some products can be rated at a frequency which is different than the account's billing frequency. For example, daily, weekly, etc.

The transaction feed management process includes the following sub-processes:

- Transaction Aggregation (which includes the Header Validation, Transaction Validation and Initial Product Determination, Product Pricing Verification, Aggregation and Clean Up sub-processes)
- Transaction Disaggregation
- Transaction Cancellation
- Transaction Rollback

You can execute each sub-process through a batch or a set of batches.

2. TFM Batch Execution Sequence

The following table indicates the sequence in which the batches must be executed while performing various sub-process in TFM:

TFM Sub-Process	Batch Sequence
Aggregation	<p>Execute the following batches in the specified order:</p> <ol style="list-style-type: none"> 1. Flush All Caches (F1-Flush) 2. Refresh Pricing (C1-TXNRP) 3. Header Validation (C1-TXNHV) 4. Transaction Validation and Initial Product Determination (C1-TXNIP) 5. Product Pricing Verification (C1-TXNVP) 6. Update Status (C1-TXNEX) 7. Service Quantity Calculation (C1-TXNSQ) 8. Mark Completion (C1-TXNCM) 9. Clean Up (C1-TXNCU) (with the Request Type parameter set to EROR) <div data-bbox="592 961 1419 1801"> <p>Note:</p> <p>The Header Validation (C1-TXNHV) batch is optional. You can directly execute the Transaction Validation and Initial Product Determination (C1-TXNIP) batch once the transactions are uploaded in the system.</p> <p>If there is any change in the TFM configuration or if there are any pricing changes while executing these batches, you need to disaggregate transactions. Once you disaggregate transactions, you must execute all above listed batches (from F1-Flush to C1-TXNCU) once again in the specified order to complete the aggregation process.</p> <p>The system allows you to execute each batch consecutively. You can execute the Transaction Validation and Initial Product Determination (C1-TXNIP) batch consecutively with the same division and same parameters, or with the different division and different parameters. But, you cannot execute the Transaction Validation and Initial Product Determination (C1-TXNIP) batch consecutively with the same division and different parameters. For example, once you execute this batch with division as D1 and transaction source as S1, you cannot execute this batch again with division as D1 and transaction source as S2 until the former transaction aggregation cycle is complete.</p> </div>

TFM Sub-Process	Batch Sequence
	<p>Note:</p> <p>Once the Product Pricing Verification (C1-TXNVP) batch is executed, you cannot execute the Transaction Validation and Initial Product Determination (C1-TXNIP) batch again with the same division and same parameters unless and until the transaction aggregation cycle is complete. Similarly, once the Service Quantity Calculation (C1-TXNSQ) batch is executed, you cannot execute the Product Pricing Verification (C1-TXNVP) batch again with the same division and same parameters unless and until the transaction aggregation cycle is complete. This rule is applicable to all subsequent batches in the transaction aggregation cycle.</p> <p>During the transaction aggregation process, you must specify the same division and same parameters across each batch. Otherwise, erroneous results might occur.</p> <p>If you have already executed the Transaction Validation and Initial Product Determination (C1-TXNIP) batch without any division, then you cannot execute this batch once again with a division unless and until the transaction aggregation cycle is complete.</p>
Rollback Transactions with the Error (EROR) or Ignored (IGNR) Status	Execute the Rollback (C1-TXNRB) batch.
Disaggregation	<p>Execute the following batches in the specified order:</p> <ol style="list-style-type: none"> 1. Disaggregation Request Creation (C1-DISTG) 2. Pending Bill Segments Deletion (C1-BSEGD) 3. Pending Bill Deletion (C1-PNBD) 4. Identify Affected Transactions (C1-IAENT) 5. Process Non-aggregated Transactions (C1-PDTXN) 6. Clean Up (C1-TXNCU) (with the Request Type parameter set to DISAGG) 7. Update Disaggregation Request Status (C1-DARSU)
Cancellation	<p>Execute the following batches in the specified order:</p> <ol style="list-style-type: none"> 1. Pending Bill Deletion (C1-DELBL) 2. Clean Up (C1-TXNCU) (with the Request Type parameter set to CNCL) 3. Cancellation (C1-TXCNC)

3. TFM Batches

This section provides detailed information about all TFM batches. It also lists and explains the parameters that you can specify while executing each TFM batch.

3.1 Rollback (C1-TXNRB)

The **Rollback (C1-TXNRB)** batch is used to rollback transactions with the **Error (EROR)** or **Ignored (IGNR)** status. Once a transaction is rolled back, the corresponding transaction legs are deleted and the status of the transaction is changed to **Uploaded (UPLD)**.

On rolling back a transaction leg whose effective pricing has the **Ignore Transaction** field set to **Yes** and the **Rating Criteria** field set to **Rate Transactions (RITX)**, the corresponding calculation lines are also deleted along with the transaction leg. If you rollback a partially disaggregated transaction which is in the **Error (EROR)** status, the transaction legs which are in the **Error (EROR)** status are only deleted and the status of the transaction is changed to **Uploaded (UPLD)**.

Note:

If you want to undertake some pre-processing activities (such as cleaning data in any custom tables) during the rollback process, you need to attach a pre-processing algorithm to the **Transaction Feed Management Rollback Preprocessing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered when you execute the **Rollback (C1-TXNRB)** batch. Note that the system invokes the algorithm which is attached on the division to which the transaction belongs.

A sample pre-processing algorithm type named **C1_ROBK_PRPC** is shipped with the product. It does not have any business logic. If you want to undertake some pre-processing activities during the rollback process, you need to create custom algorithm type and attach the respective algorithm to the **Transaction Feed Management Rollback Preprocessing** algorithm spot of the respective division. You can refer to the **C1_ROBK_PRPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

If any error occurs in the application while executing the pre-processing algorithm on a transaction, all transactions in the chunk are aborted and the subsequent chunk is considered for further processing.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The records are retrieved from the **CI_ROLLBACK_TXN_DETAIL** table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Status	Yes	Used to indicate whether you want to rollback transactions which are in the Ignored (IGNR) or Error (EROR) status. The valid values are: <ul style="list-style-type: none">IGNREROR

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to rollback transactions received through a particular feed.
Transaction Source	No	Used when you want to rollback transactions received from a particular transaction source.
Division	No	Used when you want to rollback transactions belonging to a particular division.
Rollback From Date	No	Used when you want to rollback transactions which were performed from a particular date onwards. You must specify the date in the YYYY-MM-DD format. Note: The rollback from date cannot be later than the rollback to date.
Rollback To Date	No	Used when you want to rollback transactions which were performed till a particular date. You must specify the date in the YYYY-MM-DD format. Note: The rollback to date cannot be earlier than the rollback from date.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Rollback (C1-TXNRB)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the status of the transaction which is rolled back is changed to **Uploaded (UPLD)**. The rolled back transaction is added in the CI_TXN_DETAIL_STG table and deleted from the CI_TXN_DETAIL and CI_ROLLBACK_TXN_DETAIL tables. If you roll back a transaction in the **Error (EROR)** status and which has legs in either the **Completed (COMP)** or **Ignored (IGNR)** status, the transaction legs are moved from the CI_TXN_DTL_PRITM to CI_TXN_DTL_PRITM_STG table. In addition, the errors logged against the transaction are deleted from the CI_TXN_DETAIL_EXCP table. However, if you roll back a transaction in the **Ignored (IGNR)** status, the corresponding transaction legs are deleted

from the CI_TXN_DTL_PRITM table. In addition, the corresponding calculation line details (if any) are deleted from the CI_TXN_CALC, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.

3.2 Flush All Caches (F1-Flush)

The **Flush All Caches (F1-Flush)** batch is used to clean the application cache. It is a single-threaded batch. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Thread Pool	No	Used to specify the thread pool whose cache you want to clean.

Note: If the **Flush All Caches (F1-Flush)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the cache would be cleaned completely.

3.3 Refresh Pricing (C1-TXNRP)

You can store the regular and post-processing product pricing information in the following tables and thereby improve the performance of the **Product Pricing Verification (C1-TXNVP)** and **Bill Completion (C1-BLPPR)** batches:

- CI_PRC_AGRD
- CI_PRC_PL
- CI_PRC_INH_PL

If you set the **Materialized View Used** option type of the **C1_FM** feature configuration to **true**, the system will store the regular and post-processing product pricing information in the above mentioned tables. But, if you set the **Materialized View Used** option type of the **C1_FM** feature configuration to **false**, the system will not store the regular and post-processing product pricing information in the above mentioned tables.

This mechanism to store pricing information in the above mentioned tables helps to quickly search for regular or post-processing product pricing information while executing the **Product Pricing Verification (C1-TXNVP)** or **Bill Completion (C1-BLPPR)** batch, respectively. The system stores all products' pricing information irrespective of whether it is effective or not. If the price assignment ID is not stamped on any billable charge, the system refers to the price assignment search algorithm which searches for the pricing in the original tables and not in the above mentioned tables.

If there are any pricing changes, you will have to update these tables before you execute the **Product Pricing Verification (C1-TXNVP)** or **Bill Completion (C1-BLPPR)** batch. The **Refresh Pricing (C1-TXNRP)** batch is used to update the regular and post-processing product pricing information in the above mentioned tables.

This batch is a multi-threaded batch. The multi-threading is based on person ID and chunks for multi-threading are created based on numerical distribution of person ID. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Division	No	Used when you want to update the product pricing information of accounts belonging to a particular division.
Chunk Size	Yes	Used to specify the number of persons whose regular and post-processing product pricing information you want to update in each work unit.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Refresh Pricing (C1-TXNRP)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the regular and post-processing product pricing information is updated in the CI_PRC_AGRD, CI_PRC_PL, and CI_PRC_INH_PL tables. During this process, the existing data is first removed from these tables, and then the latest information is added into these tables.

3.4 Header Validation (C1-TXNHV)

The **Header Validation (C1-TXNHV)** batch is used to validate the file or header level information of transactions. This batch is a multi-threaded batch. The chunks for multi-threading are created randomly and there is no specific logic defined for chunking in the system. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to validate a particular transaction feed.
Transaction Source	No	Used when you want to validate transaction feeds received from a particular transaction source.

Parameter Name	Mandatory (Yes or No)	Description
Checksum Validation Required	No	<p>Used to indicate whether the following should be validated:</p> <ul style="list-style-type: none"> The number of transaction records in the file matches the total transaction records in the header. The sum of transaction amount matches the total transaction amount in the header. The sum of transaction volume matches the total transaction volume in the header. <p>The valid values are:</p> <ul style="list-style-type: none"> Y N
Duplicate Check Required	No	<p>Used to indicate whether the following should be validated:</p> <ul style="list-style-type: none"> File with the same header date and external header ID is not available in the system. <p>The valid values are:</p> <ul style="list-style-type: none"> Y N <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>Note:</p> <p>If the file or feed with the same header date and external header ID is already available in the system, an error message occurs indicating that the duplicate file already exists in the system.</p> <p>The duplicate check is done only against the files or feeds which are in the Validated (VALI) status and not against the feeds which are in the Uploaded (UPLD) status.</p> </div>
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Header Validation (C1-TXNHV)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

If the file or header level information is successfully validated, the status of the feed is changed to **Validated (VALI)** and the status of all transactions in the feed remains as **Uploaded (UPLD)**. However, if header validation fails, the status of the feed and all transactions in the feed is changed to **Invalid (INVL)**. In addition, all invalid transactions are moved from the CI_TXN_DETAIL_STG to CI_TXN_DETAIL table.

Check the status of the feed in the CI_TXN_HEADER table, and the status of all transactions in the feed in the CI_TXN_DETAIL_STG or CI_TXN_DETAIL table.

3.5 Transaction Validation and Initial Product Determination (C1-TXNIP)

The **Transaction Validation and Initial Product Determination (C1-TXNIP)** batch is used to validate transaction level information and then map each transaction to:

- One or more account, division, and product combination
- One or more account, division, product and TOU (variance parameter) combination (if the multi parameter based pricing feature is disabled)
- One or more account, division, product and product parameters combination (if the multi parameter based pricing feature is enabled)

Once the initial product and product parameters mapping is complete, the product parameters to which the transaction is mapped are grouped. A set of product parameters are grouped only when the multi parameter based pricing feature is enabled. This group is then used to determine the product pricing.

A transaction leg is created for each account, division, product, and variance parameter or product parameters combination.

Note:

Based on the **Processing Date for Transaction Feed Management** characteristic defined for the division to which a transaction belongs, the system determines the rules, product parameters, pricing, and exchange rate which are effective on the transaction or batch business date. The effective product parameters, pricing, and exchange rate are determined for all account and product combinations to which the transaction is mapped. If you want to use a custom processing date for a particular account and product combination to which the transaction is mapped, you need to set the PRCS_DTX_Y_TYP output parameter. In other words, the PRCS_DTX_Y_TYP output parameter helps you to override the processing date for a particular account and product combination. However, the overridden processing date is used only while determining effective product parameters, pricing, and exchange rate.

The processing date used while executing the C1-TXNIP batch is stamped in the database. If the processing date is overridden for a particular account and product combination, the overridden processing date is stamped in the database. It is then used while executing all subsequent batches in the transaction aggregation cycle. It is also used while executing a set of batches during the transaction cancellation process.

Note:

You can aggregate transaction legs in a billable charge using additional aggregation parameters along with account ID, product, variance or product parameters, aggregation start date, and aggregation end date. The additional aggregation parameters can be stamped in the form of aggregation parameter group ID while performing post-processing activities for each transaction leg. This aggregation parameter group ID will be used in subsequent batches during the transaction aggregation cycle. If you want to perform some post-processing activities on the transaction legs, you need to attach a post-processing algorithm to the **Feed Management Product Derivation Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. Note that the system invokes the algorithm which is attached on the division to which the transaction belongs. This algorithm is triggered only for transactions which are in the **Initial Product Determined (INPD)** status. It is triggered before the product parameter group is created.

A sample post-processing algorithm type named **C1_PRDR_POPC** is shipped with the product. It does not have any business logic. If you want to undertake some post-processing activities for a transaction leg, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Product Derivation Post-Processing** algorithm spot of the respective division. You can refer to the **C1_PRDR_POPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The records are retrieved from the CI_TXN_DETAIL_STG table irrespective of the value defined corresponding to a transaction in the BO_STATUS_CD column. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Batch Business Date	No	Used to identify the transactions for which you want to perform validation and determine product. The system considers the transactions whose transaction date is less than or equal to the batch business date. Note: If you do not specify any date, the batch business date is set to the current date.
Transaction Header ID	No	Used when you want to validate and derive product for transactions received through a particular feed.
Transaction Source	No	Used when you want to validate and derive product for transactions received from a particular transaction source.
Division	No	Used when you want to validate and derive product for transactions belonging to a particular division.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.

Parameter Name	Mandatory (Yes or No)	Description
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Transaction Validation and Initial Product Determination (C1-TXNIP)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful execution of this batch, the system behaves in the following manner when a transaction is recently uploaded or reaggregated after being fully disaggregated:

- The transaction record is moved from the CI_TXN_DETAIL_STG to CI_TXN_DETAIL table.
- The status of the transaction is updated in the CI_TXN_DETAIL table. The status of the transaction can be **Invalid (INVL)**, **Initial Product Determined (INPD)**, **Error (EROR)**, or **Ignored (IGNR)**. In addition, the DISAGG_SW column corresponding to the transaction is set to N.
- If a transaction could not be successfully validated, the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table. A corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- The status of the transaction leg is updated in the CI_TXN_DTL_PRITM table. The status of the transaction leg can be **Initial Product Determined (INPD)**. In addition, the IS_DISAGG column corresponding to the transaction leg is set to N.
- The post-processing algorithm is invoked for transactions which are in the **Initial Product Determined (INPD)** status. If any error occurs while executing the post-processing algorithm, the status of the transaction is changed to **Error (EROR)** and the transaction legs are not inserted in the CI_TXN_DTL_PRITM table.
- If the multi parameter based pricing feature is enabled, a unique group ID is generated for each group and added in the CI_PRICEITEM_PARM_GRP_K table. In addition, the group ID is added in the CI_TXN_DTL_PRITM table against the corresponding transaction legs.
- If a group with a set of product parameters already exists in the system, a new group is not created. Instead, the existing group is used for determining the product pricing. If the product parameters are not derived along with the product, the group ID is set to 1 in the CI_TXN_DTL_PRITM table against the corresponding transaction legs.
- If the system could not successfully create a group ID for any product to which a transaction is mapped, the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table. A corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- By default, the aggregation parameter group ID (i.e. AGG_PARM_GRP_ID column) corresponding to each transaction leg is set to 1.

- One record is added for a set of transaction legs (which are in the **Initial Product Determined (INPD)** status and which have the same account, product, variance parameter or parameter group ID, transaction date, processing date, and aggregation parameter group ID) in the CI_TXN_DTL_PRITM_SUMMARY table. The status of the record is set to I. The SUMMARY ID is generated automatically for each record.
- If the status of the transaction is changed to **Error (EROR)** or **Ignored (IGNR)** in the CI_TXN_DETAIL table, a corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- If the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table. A generic message “Transaction is in EROR due to one or more reasons.” is added corresponding to the transaction in the CI_TXN_DETAIL table.

On successful execution of this batch, the system behaves in the following manner when a transaction is reaggregated after being partially disaggregated:

- The transaction record is moved from the CI_TXN_DETAIL_STG to CI_TXN_DETAIL table and the corresponding transaction legs are moved from the CI_TXN_DTL_PRITM_STG to CI_TXN_DTL_PRITM table.
- The status of the transaction is updated in the CI_TXN_DETAIL table. The status of the transaction can be **Initial Product Determined (INPD)**, **Error (EROR)**, **Ignored (IGNR)** or **Completed (COMP)**.
- The status of the newly created transaction leg is updated in the CI_TXN_DTL_PRITM table. The status of the transaction leg can be **Initial Product Determined (INPD)**, **Error (EROR)**, or **Ignored (IGNR)**. The IS_DISAGG column is set corresponding to each transaction leg. For newly created transaction legs, the IS_DISAGG column is set to Y. And, for all existing transaction legs, the IS_DISAGG column is set to N.
- If the multi parameter based pricing feature is enabled, a unique group ID is generated for each group. Note that the group ID is generated only for the transaction legs which are newly created while executing this batch. This group ID is then added in the CI_PRICEITEM_PARM_GRP_K table. It is also added in the CI_TXN_DTL_PRITM table against the corresponding transaction legs.
- If a group with a set of product parameters already exists in the system, a new group is not created. Instead, the existing group is used for determining the product pricing. If the product parameters are not derived along with the product, the group ID is set to 1 in the CI_TXN_DTL_PRITM table against the corresponding transaction legs which are newly created while executing this batch.
- If the system could not successfully create a group ID for newly derived product to which a transaction is mapped, the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table. A corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- By default, the aggregation parameter group ID (i.e. AGG_PARM_GRP_ID column) corresponding to each transaction leg is set to 1.

- One record is added for a set of transaction legs (which are in the **Initial Product Determined (INPD)** status and which have the same account, product, variance parameter or parameter group ID, transaction date, processing date, and aggregation parameter group ID) in the CI_TXN_DTL_PRITM_SUMMARY table. The status of the record is set to I. The SUMMARY ID is generated automatically for each record.
- If the status of the transaction is changed to **Error (EROR)** or **Ignored (IGNR)** in the CI_TXN_DETAIL table, a corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- If the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table. A generic message “Transaction is in EROR due to one or more reasons.” is added corresponding to the transaction in the CI_TXN_DETAIL table.

Once you execute this batch, we recommend you to generate complete statistics for the following tables:

- CI_TXN_DETAIL
- CI_TXN_DTL_PRITM
- CI_ROLLBACK_TXN_DETAIL
- CI_PRICEITEM_PARM_GRP_K
- CI_TXN_DTL_PRITM_SUMMARY
- CI_AGG_PARM_GRP_DTL

You must execute the following statements to gather statistics for the above mentioned tables:

BEGIN

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM',          TABNAME=>'CI_TXN_DETAIL',
GRANULARITY=>'ALL',  CASCADE=>TRUE,  METHOD_OPT=>  'FOR ALL COLUMNS SIZE AUTO',
DEGREE=>32);
```

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM',          TABNAME=>'CI_TXN_DTL_PRITM',
GRANULARITY=>'ALL',  CASCADE=>TRUE,  METHOD_OPT=>  'FOR ALL COLUMNS SIZE AUTO',
DEGREE=>32);
```

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM', TABNAME=>'CI_ROLLBACK_TXN_DETAIL',
GRANULARITY=>'ALL',  CASCADE=>TRUE,  METHOD_OPT=>  'FOR ALL COLUMNS SIZE AUTO',
DEGREE=>32);
```

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM',
TABNAME=>'CI_PRICEITEM_PARM_GRP_K', GRANULARITY=>'ALL', CASCADE=>TRUE, METHOD_OPT=>
'FOR ALL COLUMNS SIZE AUTO', DEGREE=>32);
```

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM',
TABNAME=>'CI_TXN_DTL_PRITM_SUMMARY',          GRANULARITY=>'ALL',          CASCADE=>TRUE,
METHOD_OPT=> 'FOR ALL COLUMNS SIZE AUTO', DEGREE=>32);
```

```
DBMS_STATS.GATHER_TABLE_STATS(OWNNAME=>'CISADM', TABNAME=>'CI_AGG_PARM_GRP_DTL',
GRANULARITY=>'ALL',  CASCADE=>TRUE,  METHOD_OPT=>  'FOR ALL COLUMNS SIZE AUTO',
DEGREE=>32);
```

END;

3.6 Product Pricing Verification (C1-TXNVP)

The **Product Pricing Verification (C1-TXNVP)** batch is used to check whether effective pricing is available for:

- An account, division, product and/or TOU combination on the processing date if the multi parameter based pricing feature is disabled
- An account, division, product and/or product parameters (parameter group) combination on the processing date if the multi parameter based pricing feature is enabled

Note:

The processing date which is stamped against a transaction leg is used to determine effective pricing for the transaction leg.

If you want to perform some post-processing activities on a summary record in the CI_TXN_DTL_PRITM_SUMMARY table, you need to attach a post-processing algorithm to the **Feed Management Verify Pricing Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered once the effective pricing is determined for a transaction leg. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs.

A sample post-processing algorithm type named **C1-VRPR_POPC** is shipped with the product. It does not have any business logic. If you want to undertake some post-processing activities on a summary record, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Verify Pricing Post-Processing** algorithm spot of the respective division. You can refer to the **C1-VRPR_POPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

If any error occurs in the application while executing the post-processing algorithm on a summary record, all summary records in the chunk are aborted and the subsequent chunk is considered for further processing.

This batch is a multi-threaded batch. The multi-threading is based on summary ID and chunks for multi-threading are created based on numerical distribution of summary ID. The records with the **I** status are retrieved from the CI_TXN_DTL_PRITM_SUMMARY table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to verify product pricing for transactions received through a particular feed.
Transaction Source	No	Used when you want to verify product pricing for transactions received from a particular transaction source.
Division	No	Used when you want to verify product pricing for transactions belonging to a particular division.

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Product Pricing Verification (C1- TXNVP)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful execution of this batch, the pricing and aggregation information is stored in the summary record in the CI_TXN_DTL_PRITM_SUMMARY table. In addition, the status of the summary record is set to NULL. If the multi parameter based pricing feature is enabled, the summary record contains pricing and aggregation information for account, product, parameter group ID, transaction date, processing date, and aggregation parameter group ID combination. If the multi parameter based pricing feature is disabled, the summary record contains pricing and aggregation information for account, product, variance parameter, transaction date, processing date, and aggregation parameter group ID combination.

3.7 Update Status (C1-TXNEX)

The **Update Status (C1-TXNEX)** batch is used to update the status of the transaction and its transaction legs. If a transaction leg is ignored and not considered for billing, the status of the transaction leg is changed to **Ignored (IGNR)**, whereas the status of the transaction remains as **Initial Product Determined (INPD)**. However, if all legs of a transaction are ignored and not considered for billing, the status of the transaction and transaction legs is changed to **Ignored (IGNR)**. And, if the effective pricing is not available for one or more products to which a transaction is mapped, the status of the corresponding transaction leg and transaction is changed to **Error (EROR)**.

In addition, the status of the transaction and transaction leg is changed to **Error (EROR)** when:

- There is no contract available with the specified contract type on the transaction date or when the contract is inactive.
- There are multiple effective contracts of the same contract type (available on the transaction date) in **Active**, **Pending Stop**, or **Stop** status.
- The **Price Assignment Search** algorithm is not defined for the division.
- The parameter values are either not defined or invalid in the **Price Assignment Search** algorithm on the processing date.
- The period in which the transaction date falls is not defined in the aggregation schedule.

Besides updating the status, the **Update Status (C1-TXNEX)** batch determines the rate for transaction legs whose effective pricing has either of the following set of attributes:

- **Ignore Transaction** is set to **Yes** and **Rating Criteria** is set to **Rate Transactions (RITX)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**

Each set of pricing attributes indicates how the transaction legs must be rated before billing. For more information about the different ways in which a transaction leg can be rated, refer to the Transaction Rating Before Billing section in *Oracle Revenue Management and Billing Banking User Guide*.

Note: If you want to do some pre-processing activities before invoking the rates engine, you need to attach a pre-processing algorithm to the **Feed Management Rate Pre-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs. A sample pre-processing algorithm type named **C1_RTCL_PRPC** is shipped with the product. It does not have any business logic. If you want to undertake some pre-processing activities (such as passing additional parameters (for example, additional characteristics or SQLs) to rates engine) before invoking the rates engine, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Rate Pre-Processing** algorithm spot of the respective division. You can refer to the **C1_RTCL_PRPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

Once the rate is determined for transaction legs, a set of rate component characteristics (with the characteristic entity set to **Billable Charge Line**) and their values on the calculation line are grouped. These groups are used for accumulating pre-calculated charges. A unique aggregation parameter group ID is generated for each group. If a group with a set of rate component characteristics and their values already exists in the system, a new group is not created. Instead, the existing group is used for accumulating pre-calculated charges. The aggregation parameter group ID is created only when you attach an algorithm of the **C1_RTCL_POPC** algorithm type to the **Feed Management Rate Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The transactions which are in the **Initial Product Determined (INPD)** status are retrieved from the CI_TXN_DETAIL table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to change the status of transactions received through a particular feed.
Transaction Source	No	Used when you want to change the status of transactions received from a particular transaction source.
Division	No	Used when you want to change the status of transactions belonging to a particular division.

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. <div>Note: By default, the thread count is set to 0.</div>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Update Status (C1-TXNEX)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful execution of this batch, the system behaves in the following manner when a transaction is recently uploaded or reaggregated after being fully disaggregated:

- If the transaction is not considered for billing for all products (to which it is mapped), the status of the transaction is changed to **Ignored (IGNR)** in the CI_TXN_DETAIL table. However, if the transaction is considered for billing for one or more products, but not for all products (to which it is mapped), the status of the transaction remains as **Initial Product Determined (INPD)** in the CI_TXN_DETAIL table.
- If a transaction leg is not considered for billing, the status of the transaction leg is changed to **Ignored (IGNR)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria is stored in the TXN_RATING_CRITERIA column corresponding to the transaction leg. If the **Ignore Transaction** field is set to **Yes** and the **Rating Criteria** field is set to **Rate Transactions (RITX)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.
However, if a transaction leg is considered for billing, the status of the transaction leg remains as **Initial Product Determined (INPD)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria and aggregate transaction switch is stored in the TXN_RATING_CRITERIA and DO_NOT_AGG_SW columns, respectively, corresponding to the transaction leg. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.
- If the effective pricing could not be determined for one or more products or for all products (to which the transaction is mapped), the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table.

- If the effective pricing could not be determined for a product, the status of the corresponding transaction leg is changed to **Error (EROR)** in the CI_TXN_DTL_PRITM table. However, if effective pricing is determined for a product, the status of the corresponding transaction leg remains as **Initial Product Determined (INPD)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria and aggregate transaction switch is stored in the TXN_RATING_CRITERIA and DO_NOT_AGG_SW columns, respectively, corresponding to the transaction leg. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.
- If the calculation lines are generated for a transaction leg, the corresponding details are stored in the CI_TXN_CALC, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.
- The aggregation parameter group ID is generated and stored in the AGG_PARM_GRP_ID column corresponding to the transaction leg in the CI_TXN_DTL_PRITM table. The corresponding details are stored in the CI_AGG_PARM_GRP_DTL table.
- The rates engine is invoked for the transaction legs in the order in which the transaction legs are derived. If the rate calculation fails for a transaction leg due to some reason, the rates engine is not invoked for the remaining legs in the transaction. The status of the transaction is changed to **Error (EROR)** and calculation details are not inserted for any transaction leg in the database.
- If the status of the transaction is changed to **Error (EROR)** or **Ignored (IGNR)** in the CI_TXN_DETAIL table, a corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- If the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table. A generic message “One or more Sub Transactions are in EROR state.” is added corresponding to the transaction in the CI_TXN_DETAIL table.

On successful execution of this batch, the system behaves in the following manner when a transaction is reaggregated after being partially disaggregated:

- If a transaction leg is not considered for billing, the status of the transaction leg is changed to **Ignored (IGNR)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria is stored in the TXN_RATING_CRITERIA column corresponding to the transaction leg. If the **Ignore Transaction** field is set to **Yes** and the **Rating Criteria** field is set to **Rate Transactions (RITX)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.

However, if a transaction leg is considered for billing, the status of the transaction leg remains as **Initial Product Determined (INPD)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria and aggregate transaction switch is stored in the TXN_RATING_CRITERIA and DO_NOT_AGG_SW columns, respectively, corresponding to the transaction leg. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.

- If the effective pricing could not be determined for a product, the status of the corresponding transaction leg is changed to **Error (EROR)** in the CI_TXN_DTL_PRITM table. However, if effective pricing is determined for a product, the status of the corresponding transaction leg remains as **Initial Product Determined (INPD)** in the CI_TXN_DTL_PRITM table. In addition, the rating criteria and aggregate transaction switch is stored in the TXN_RATING_CRITERIA and DO_NOT_AGG_SW columns, respectively, corresponding to the transaction leg. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column.
- If the calculation lines are generated for a transaction leg, the corresponding details are stored in the CI_TXN_CALC, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.
- The aggregation parameter group ID is generated and stored in the AGG_PARM_GRP_ID column corresponding to the transaction leg in the CI_TXN_DTL_PRITM table. The corresponding details are stored in the CI_AGG_PARM_GRP_DTL table.
- The rates engine is invoked for the transaction legs in the order in which the transaction legs are derived. If the rate calculation fails for a transaction leg due to some reason, the rates engine is not invoked for the remaining legs in the transaction. The status of the transaction is changed to **Error (EROR)** and calculation details are not inserted for any transaction leg in the database.
- If the status of the transaction is changed to **Error (EROR)** or **Ignored (IGNR)** in the CI_TXN_DETAIL table, a corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table.
- If the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table. A generic message “One or more Sub Transactions are in EROR state.” is added corresponding to the transaction in the CI_TXN_DETAIL table.
- If any error occurs in the application while executing the pre-processing and post-processing algorithms on a transaction leg, the respective transaction in the chunk is aborted and the subsequent transactions in the chunk are considered for further processing. The error details are stored corresponding to the transaction leg in the CI_TXN_DTL_PRITM table.

3.8 Service Quantity Calculation (C1-TXNSQ)

The **Service Quantity Calculation (C1-TXNSQ)** batch is used to aggregate the transaction legs, create billable charges, and then update the SQI values in the billable charges. Once the SQI values are updated in the billable charge, the rate is determined for the transaction leg whose effective pricing has either of the following set of attributes:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Aggregate transactions and then rate aggregated SQs (AGTR)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **No**, and **Rating Criteria** is set to **Rate Transactions (RITX)**

Each set of pricing attributes indicates how the transaction legs must be rated before billing. For more information about the different ways in which a transaction leg can be rated, refer to the Transaction Rating Before Billing section in *Oracle Revenue Management and Billing Banking User Guide*.

Note:

If you want to perform some post-processing activities on a billable charge, you need to attach a post-processing algorithm to the **Feed Management Billable Charge Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered once the billable charge is created and SQIs are updated in the billable charge. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs. A sample post-processing algorithm type named **C1_BCHG_POPC** is shipped with the product. It does not have any business logic. If you want to undertake some post-processing activities on a billable charge, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Billable Charge Post-Processing** algorithm spot of the respective division. You can refer to the **C1_BCHG_POPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

If you want to do some pre-processing activities before invoking the rates engine, you need to attach a pre-processing algorithm to the **Feed Management Rate Pre-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs. A sample pre-processing algorithm type named **C1_RTCL_PRPC** is shipped with the product. It does not have any business logic. If you want to undertake some pre-processing activities (such as passing additional parameters (for example, additional characteristics or SQIs) to rates engine) before invoking the rates engine, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Rate Pre-Processing** algorithm spot of the respective division. You can refer to the **C1_RTCL_PRPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

Once the rate is determined for transaction legs, a set of rate component characteristics (with the characteristic entity set to **Billable Charge Line**) and their values on the calculation line are grouped. These groups are used for accumulating pre-calculated charges. A unique aggregation parameter group ID is generated for each group. If a group with a set of rate component characteristics and their values already exists in the system, a new group is not created. Instead, the existing group is used for accumulating pre-calculated charges. The aggregation parameter group ID is created only when you attach an algorithm of the **C1_RTCL_POPC** algorithm type to the **Feed Management Rate Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs.

This batch is a multi-threaded batch. For non-aggregated transaction legs, the multi-threading is based on distinct transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The transaction legs which are in the **Initial Product Determined (INPD)** status are retrieved from the CI_TXN_DETAIL_PRITM table. However, for aggregated transaction legs, the multi-threading is based on summary ID and chunks for multi-threading are created based on numerical distribution of summary ID. The records are retrieved from the CI_TXN_DTL_PRITM_SUMMARY table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to create billable charges for transactions received through a particular feed.

Parameter Name	Mandatory (Yes or No)	Description
Transaction Source	No	Used when you want to create billable charges for transactions received from a particular transaction source.
Division	No	Used when you want to create billable charges for transactions belonging to a particular division.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. <div>Note: By default, the thread count is set to 0.</div>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note:

Once the **Service Quantity Calculation (C1-TXNSQ)** batch is executed, you must execute the **Mark Completion (C1-TXNCM)** and **Clean Up (C1_TXNCU)** batches. Even if the **Service Quantity Calculation (C1-TXNSQ)** batch fails, you must execute the **Mark Completion (C1-TXNCM)** and **Clean Up (C1_TXNCU)** batches. The **Clean Up (C1_TXNCU)** batch must be executed with the **Request Type** parameter set to **EROR**.

If the **Service Quantity Calculation (C1-TXNSQ)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, billable charges are created and added in the CI_BILL_CHG and CI_BILL_CHG_K tables. The corresponding SQIs and characteristics are added in the CI_BCHG_SQ and CI_BILL_CHG_CHAR tables, respectively. The billable charge pass through lines and their characteristics are added in the CI_B_CHG_LINE and CI_B_LN_CHAR tables, respectively.

If the **Aggregate Transaction** field is set to **No**, the billable charge ID is updated corresponding to the transaction leg in the CI_TXN_DTL_PRITM table. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **No**, and the **Rating Criteria** field is set to **Rate Transactions (RITX)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column corresponding to the transaction leg in the CI_TXN_DTL_PRITM table.

However, if the **Aggregate Transaction** field is set to **Yes**, the billable charge ID is updated corresponding to the record in the CI_TXN_DTL_PRITM_SUMMARY table. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Aggregate transactions and then rate aggregated SQs (AGTR)**, the calculation lines are generated and the transaction calculation ID is stored in the TXN_CALC_ID column corresponding to the record in the CI_TXN_DTL_PRITM_SUMMARY table.

In addition, if the **Aggregate Transaction** field is set to **Yes**, the status of the records is updated in the CI_TXN_DTL_PRITM_SUMMARY table. If the aggregated billable charge is created and updated successfully, the status of the record is changed to **C**. However, if any error occurs while creating or updating the aggregated billable charge or while executing the pre-processing and post-processing algorithms on aggregated transaction legs, the status of the record is changed to **E**. The errors are logged against the summary record in the CI_TXN_DTL_PRITM_SUMMARY table. If any error occurs while creating or updating the non-aggregated billable charge or while executing the pre-processing and post-processing algorithms on non-aggregated transaction legs, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table.

If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **No**, and the **Rating Criteria** field is set to **Rate Transactions (RITX)**, the rates engine is invoked for the transaction legs in the order in which the transaction legs are derived. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Aggregate transactions and then rate aggregated SQs (AGTR)**, the rates engine is invoked for aggregated service quantities.

The pass through billable charge line is created by accumulating calculation lines which have the same distribution code, currency code, description on bill, aggregation parameter group ID (which is created based on the rate component characteristics) combination.

If the calculation lines are generated for a transaction leg, the corresponding details are stored in the CI_TXN_CALC, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables. If you re-execute the **Service Quantity Calculation (C1-TXNSQ)** batch, the calculation lines and pass through billable charge lines will be updated, accordingly.

3.9 Mark Completion (C1-TXNCM)

The **Mark Completion (C1-TXNCM)** batch is used to update the status of the transaction and its transaction legs. If the SQI values are updated successfully in the billable charge, the status of the transaction leg is changed to **Completed (COMP)**. But, if the SQIs are not defined for the product — division combination, the transaction aggregation rule is not defined for the SQI, or the exchange rate is not available during currency conversion, the status of the transaction leg is changed to **Error (EROR)**. If all legs of a transaction are in the **Completed (COMP)** status, the status of the transaction is changed to **Completed (COMP)**. But, if any of the transaction leg is in the **Error (EROR)** status, the status of the transaction is changed to **Error (EROR)**.

Besides updating the status, the **Mark Completion (C1-TXNCM)** batch does the following with the remaining legs when billable charge is not created for a transaction leg:

Rate for other transaction leg is determined using the following set of pricing attributes...	Then....
Ignore Transaction is set to Yes and Rating Criteria is set to Rate Transactions (RITX)	The corresponding calculation lines of the transaction leg are deleted.

Rate for other transaction leg is determined using the following set of pricing attributes...	Then....
Ignore Transaction is set to No , Aggregate Transaction is set to No , and Rating Criteria is set to Rate Transactions (RITX)	The corresponding billable charge and calculation lines of the transaction leg are deleted.
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Rate individual transactions and aggregate calc lines across transactions (RITA)	The corresponding calculation lines of the transaction leg are deleted.
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Aggregate transactions and then rate aggregated SQs (AGTR)	The corresponding billable charge and calculation lines of the transaction leg are not deleted.
Ignore Transaction is set to No , Aggregate Transaction is set to No , and Rating Criteria is set to Do Not Rate Transactions	The corresponding non-aggregated billable charge is deleted.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The transactions which are in the **Initial Product Determined (INPD)** status are retrieved from the CI_TXN_DETAIL table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to change the status of transactions received through a particular feed.
Transaction Source	No	Used when you want to change the status of transactions received from a particular transaction source.
Division	No	Used when you want to change the status of transactions belonging to a particular division.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Mark Completion (C1-TXNCM)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, check the status of the transaction and transaction legs in the CI_TXN_DETAIL and CI_TXN_DTL_PRITM tables, respectively.

If the billable charge is successfully created and updated for all products to which the transaction is mapped, the status of the transaction and its legs is changed to **Completed (COMP)** in CI_TXN_DETAIL and CI_TXN_DTL_PRITM tables, respectively. However, if any error occurs while creating or updating the billable charge for any product (to which the transaction is mapped), the status of the transaction and the corresponding transaction leg is changed to **Error (EROR)**. In addition, if a billable charge is created for any non-aggregated transaction leg in the transaction whose status is changed to **Error (EROR)**, then the corresponding billable charge record is deleted from the CI_BILL_CHG, CI_BILL_CHG_K, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables. And, the corresponding calculation records are deleted from CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables when effective pricing of the transaction leg has either of the following set of attributes:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **No**, and **Rating Criteria** is set to **Rate Transactions (RITX)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**

If the status of the transaction is changed to **Error (EROR)** in the CI_TXN_DETAIL table, a corresponding transaction entry is added in the CI_ROLLBACK_TXN_DETAIL table. In addition, the errors are logged against the transaction in the CI_TXN_DETAIL_EXCP table. A generic message "One or more Sub Transactions are in EROR state." is added corresponding to the transaction in the CI_TXN_DETAIL table. However, if an error occurs while creating aggregated billable charge, a generic message "One or more Sub Transactions are in EROR state." is added corresponding to the record in the CI_TXN_DTL_PRITM_SUMMARY table.

3.10 Clean Up (C1-TXNCU)

The **Clean Up (C1-TXNCU)** batch is used to recalculate SQIs in the aggregated billable charge or delete the aggregated billable charge depending on whether the aggregated billable charge includes transaction legs in the **Completed (COMP)** status.

When an aggregated billable charge includes the transaction legs which are in the **Error (EROR)** and **Completed (COMP)** statuses, the **Clean Up (C1-TXNCU)** batch does the following:

Billable charge contains a transaction leg with the following set of pricing attributes...	Then....
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Do Not Rate Transactions	The SQIs are recalculated in the billable charge.

Billable charge contains a transaction leg with the following set of pricing attributes...	Then....
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Aggregate transactions and then rate aggregated SQs (AGTR)	The SQIs are recalculated in the billable charge and the rate is determined for aggregated service quantities. Once the rate is determined, pass through charges are calculated and accumulated in a pass through billable charge line based on the distribution code, currency code, description on bill, and aggregation parameter group ID combination.
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Rate individual transactions and aggregate calc lines across transactions (RITA)	The pass through charges are recalculated and accumulated accordingly.

However, when the transaction legs in the **Error (EROR)** status are only aggregated in a billable charge, the **Clean Up (C1-TXNCU)** batch does the following:

Billable charge contains a transaction leg with the following set of pricing attributes...	Then....
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Do Not Rate Transactions	The aggregated billable charge is deleted.
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Aggregate transactions and then rate aggregated SQs (AGTR)	The aggregated billable charge and the corresponding calculation lines are deleted.
Ignore Transaction is set to No , Aggregate Transaction is set to Yes , and Rating Criteria is set to Rate individual transactions and aggregate calc lines across transactions (RITA)	The aggregated billable charge is deleted.

Besides the aggregation process, this batch is also used during the following sub-processes:

- **Cancellation** - During the cancellation process, it deletes or cancels non-aggregated billable charges and recalculates SQIs in aggregated billable charges. If an aggregated billable charge has transaction legs from only one transaction feed which is cancelled, then the aggregated billable charge is deleted or cancelled. If the aggregated or non-aggregated billable charge is deleted or cancelled, the corresponding calculation lines (if any) are also deleted.
- **Disaggregation** - During the disaggregation process, it deletes or cancels an aggregated billable charge when all the corresponding transaction legs which were aggregated in the billable charge are deleted during disaggregation. In addition, the corresponding calculation lines (if any) are deleted.

Note:

The SQIs in an aggregated billable charge are recalculated only when the **SQ Recalculation Required** option type in the **C1_FM** feature configuration is set to **Y**. If you set the **SQ Recalculation Required** option type in the **C1_FM** feature configuration to **N**, the SQIs are not recalculated in an aggregated billable charge. We recommend you recalculate SQIs in an aggregated billable charge when more than one account bears the charges for a transaction.

If you want to perform some post-processing activities on a billable charge, you need to attach a post-processing algorithm to the **Feed Management Billable Charge Post-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered once the billable charge is created and SQIs are updated in the billable charge. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs. A sample post-processing algorithm type named **C1_BCHG_POPC** is shipped with the product. It does not have any business logic. If you want to undertake some post-processing activities on a billable charge, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Billable Charge Post-Processing** algorithm spot of the respective division. You can refer to the **C1_BCHG_POPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

This batch is a multi-threaded batch. The multi-threading and chunking logic is different for each request type:

- **EROR** – For aggregated transactions, the multi-threading is based on summary ID and chunks for multi-threading are created based on numerical distribution of summary ID. The records with the status **C** are retrieved from the CI_TXN_DTL_PRITM_SUMMARY table.
- **CNCL** - The multi-threading is based on distinct billable charge ID and chunks for multi-threading are created based on numerical distribution of billable charge ID. The records are retrieved from the CI_TXN_DTL_PRITM and CI_TXN_DTL_PRITM_STG tables. This batch creates separate chunks for aggregated and non-aggregated billable charges.
- **DISAGG** - The multi-threading is based on billable charge ID and chunks for multi-threading are created based on numerical distribution of billable charge ID. The aggregated billable charges are retrieved from the CI_DISAGG_BCHG_DETAIL table.

You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	Yes (Conditional)	Used when you want to delete or update billable charges created for transactions received through a particular feed. <div> Note: This parameter should not be used during the disaggregation process. This parameter is required when you set the request type to CNCL. </div>

Parameter Name	Mandatory (Yes or No)	Description
Transaction Source	No	Used when you want to delete or update billable charges created for transactions received from a particular transaction source. Note: This parameter should not be used during the cancellation and disaggregation processes.
Division	No	Used when you want to delete or update billable charges created for transactions belonging to a particular division. Note: This parameter should not be used during the cancellation process.
Account ID	No	Used when you want to delete or update billable charges of a particular account. Note: This parameter should be used only during the disaggregation process.
Bill Cycle	No	Used when you want to delete or update billable charges of accounts having a particular bill cycle. Note: This parameter should be used only during the disaggregation process.
Request Type	Yes	Used to indicate the process during which you want to execute the batch. The valid values are: <ul style="list-style-type: none"> • CNCL • EROR • DISAGG
Disaggregate Transactions From Date	Yes (Conditional)	Used when you want to delete or update billable charges created for transactions which were performed from a particular date onwards. You must specify the date in the YYYY-MM-DD format. Note: This parameter should be used only during the disaggregation process. This parameter is required when you set the request type to DISAGG .
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.

Parameter Name	Mandatory (Yes or No)	Description
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Clean Up (C1-TXNCU)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, billable charge records are either updated or deleted from the CI_BILL_CHG, CI_BILL_CHG_K, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables. During the aggregation process, the records are deleted from the CI_TXN_DTL_PRITM_SUMMARY table. If the **Ignore Transaction** field is set to **No**, the **Aggregate Transaction** field is set to **Yes**, and the **Rating Criteria** field is set to **Aggregate transactions and then rate aggregated SQs (AGTR)**, the calculation records are either updated or deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.

During the cancellation process, the non-aggregated billable charge records are deleted from the CI_BILL_CHG, CI_BILL_CHG_K, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables. The corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables when effective pricing of the transaction leg has either of the following set of attributes:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **No**, and **Rating Criteria** is set to **Rate Transactions (RITX)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**

If an aggregated billable charge includes legs of transactions from more than one feed, the SQIs and pass through lines are recalculated in the aggregated billable charge. The corresponding calculation records are updated in the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables when effective pricing of the transaction leg has the following set of attributes:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Aggregate transactions and then rate aggregated SQs (AGTR)**

But, if an aggregated billable charge includes legs of transactions from a feed which you want to cancel, the aggregated billable charges are deleted from the CI_BILL_CHG, CI_BILL_CHG_K, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables. The corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.

If the aggregated or non-aggregated billable charge is in the **Canceled** status, the corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables. However, if any bill segment is in the **Canceled** status, then:

- The status of the corresponding billable charge is changed to **Canceled**.
- The corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.

During the disaggregation process, if the **BILLABLE_CHG_ACT_CD** column corresponding to the billable charges in the CI_DISAGG_BCHG_DETAIL table is set to **DELETE (10)**, the aggregated billable charge records are deleted from the CI_BILL_CHG_K, CI_BILL_CHG, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables. However, if the **BILLABLE_CHG_ACT_CD** column corresponding to the billable charges in the CI_DISAGG_BCHG_DETAIL table is set to **CANCEL (20)**, the status of the aggregated billable charges is changed to **Canceled** in the CI_BILL_CHG table. In addition, the **BO_STATUS_CD** column corresponding to the billable charges in the CI_DISAGG_BCHG_DETAIL table is set to **C**. Irrespective of whether the aggregated billable charges are deleted or cancelled, the corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables.

3.11 Disaggregation Request Creation (C1-DISTG)

The **Disaggregation Request Creation (C1-DISTG)** batch is used to create disaggregation request for an account and customer. When you create a disaggregation request for an account, the transactions mapped to the account are disaggregated. However, when you create a disaggregation request for a customer, the transactions mapped to the customer's accounts and its child customers' accounts are disaggregated. This disaggregation request is added in the CI_TXN_DISAGG_REQ table. At present, the system disaggregates transactions at the account level and not at the product level. Let us understand this with the help of an example. The following table lists the accounts and products to which T1 is mapped:

Transaction	Account	Product
T1	A1	P1
T1	A1	P2
T1	A2	P1
T1	A2	P2

Now, if the pricing of P1 for A1 changes, the system creates a disaggregation request for A1 and identifies all transaction legs of A1 for disaggregation. In this example, the system will consider the first two transaction legs - T1-A1-P1 and T1-A1-P2 - for disaggregation even if the pricing of P2 for A1 has not changed.

This batch is a multi-threaded batch. The multi-threading is based on division and chunks for multi-threading are created based on numerical distribution of division. The records are retrieved from the CI_TXN_DTL_PRITM and CI_TXN_DTL_PRITM_STG tables. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Division	No	Used when you want to create disaggregation request for accounts belonging to a particular division.
Bill Cycle	No	Used when you want to create disaggregation request for accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	Used when you want to create disaggregation request for accounts for which transactions were performed from a particular date onwards. You must specify the date in the YYYY-MM-DD format.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Disaggregation Request Creation (C1-DISTG)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, disaggregation requests for accounts and customers are added in the CI_TXN_DISAGG_REQ table. In addition, the BO_STATUS_CD column corresponding to the disaggregation request in the CI_TXN_DISAGG_REQ table is set to **PENDING**.

Note that the disaggregation request is added only when there is no pending disaggregation request available for the account or customer in the CI_TXN_DISAGG_REQ table.

3.12 Pending Bill Segments Deletion (C1-BSEGD)

The **Pending Bill Segments Deletion (C1-BSEGD)** batch is used to delete bill segments which are in the **Freezable** or **Error** status. It does not delete bill segments which are in the **Pending Cancel**, **Frozen**, or **Canceled** status.

This batch is a multi-threaded batch. The multi-threading is based on bill segment ID and chunks for multi-threading are created based on numerical distribution of bill segment ID. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Account ID	No	Used to indicate the account whose bill segments you want to delete.

Parameter Name	Mandatory (Yes or No)	Description
Division	No	Used when you want to delete bill segments of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to delete bill segments of accounts having a particular bill cycle.
Request Type	No	<p>Used to indicate whether you want to delete bill segments during the disaggregation process. The valid values are:</p> <ul style="list-style-type: none"> • BILLING – When you select this option, the system deletes bill segments of pending bills of accounts that meet the criteria. • DISAGG – When you select this option, the system deletes bill segments of accounts that meet the criteria and for which the disaggregation request is created. <p>Note: If you do not specify any value, by default, the parameter value is set to BILLING.</p>
Bill Type	No	<p>Used to indicate whether you want to delete bill segments of pending bills which are created either manually (i.e. through the user interface) or through the batch process. The valid values are:</p> <ul style="list-style-type: none"> • ALL • MANL • BATCH <p>Note: If you do not specify any value, by default, the parameter value is set to ALL.</p>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note:

At present, the batch business date is not used (or considered) while executing the **Pending Bill Segments Deletion (C1-BSEGD)** batch.

If the **Pending Bill Segments Deletion (C1-BSEGD)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the bill segments in the **Freezable** or **Error** status and their corresponding financial transactions are deleted. The corresponding bill ID is added in the CI_DELETE_BILL_DETAIL table for further processing.

3.13 Pending Bill Deletion (C1-PNBD)

The **Pending Bill Deletion (C1-PNBD)** batch checks whether the pending bills listed in the CI_DELETE_BILL_DETAIL table have any bill segments in the **Pending Cancel**, **Frozen**, or **Canceled** status. If there are bill segments in the **Pending Cancel**, **Frozen**, or **Canceled** status, the pending bill is not deleted. However, if there are no bill segments in the **Pending Cancel**, **Frozen**, or **Canceled** status, the pending bill is deleted. This batch deletes regular pending bills and not adhoc pending bills.

Note: If the pending bills have bill segments, you must first execute the **Pending Bill Segments Deletion (C1-BSEGD)** batch and then execute the **Pending Bill Deletion (C1-PNBD)** batch. While executing these batches in the specified order, ensure that you specify the same parameters in both these batches.

This batch is a multi-threaded batch. The multi-threading is based on bill ID and chunks for multi-threading are created based on numerical distribution of bill ID. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. <div>Note: By default, the thread count is set to 0.</div>
Account ID	No	Used to indicate the account whose pending bills you want to delete.
Division	No	Used when you want to delete pending bills of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to delete pending bills of accounts having a particular bill cycle.
Request Type	No	Used to indicate whether you want to delete pending bills during the disaggregation process. The valid values are: <ul style="list-style-type: none"> BILLING – When you select this option, the system deletes pending bills of accounts that meet the criteria. DISAGG – When you select this option, the system deletes pending bills of accounts that meet the criteria and for which the disaggregation request is created. <div>Note: If you do not specify any value, by default, the parameter value is set to BILLING.</div>

Parameter Name	Mandatory (Yes or No)	Description
Bill Type	No	<p>Used to indicate whether you want to delete pending bills which are created either manually (i.e. through the user interface) or through the batch process. The valid values are:</p> <ul style="list-style-type: none"> • ALL • MANL • BATCH <p>Note: If you do not specify any value, by default, the parameter value is set to ALL.</p>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note:

At present, the batch business date is not used (or considered) while executing the **Pending Bill Deletion (C1-PNBD)** batch.

If the **Pending Bill Deletion (C1-PNBD)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the BILL_ACT_CD column corresponding to a pending bill in the CI_DELETE_BILL_DETAIL table is set to either 10 or 20. 10 indicate that the pending bill contains no bill segments in the **Pending Cancel**, **Frozen**, or **Canceled** status, and therefore can be deleted. However, 20 indicate that the pending bill contains bill segments in the **Pending Cancel**, **Frozen**, or **Canceled** status, and therefore cannot be deleted. If the BILL_ACT_CD column corresponding to a pending bill is set to 10, the system deletes the pending bill.

3.14 Identify Affected Transactions (C1-IAENT)

The **Identify Affected Transactions (C1-IAENT)** batch is used to fetch disaggregation requests from the CI_TXN_DISAGG_REQ table. It identifies the transactions and the corresponding aggregated and non-aggregated billable charges for disaggregation. If the bill segment of a billable charge is in the **Pending Cancel** or **Frozen** status, the system will not identify the billable charge for deletion.

Pre-requisites:

- A disaggregation request for account or customer must be present in the CI_TXN_DISAGG_REQ table.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The records are retrieved from the CI_TXN_DTL_PRITM and CI_TXN_DTL_PRITM_STG tables. You can specify either of the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Account ID	No	Used when you want to identify transactions of a particular account.
Division	No	Used when you want to identify transactions of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to identify transactions of accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	Used when you want to identify transactions which were performed from a particular date onwards. You must specify the date in the YYYY-MM-DD format. Note: The aggregated billable charge (which is affected) must not include a transaction leg whose transaction date is earlier than the disaggregation transactions from date. Otherwise, erroneous results will occur. Therefore, ensure that you specify the appropriate value for the Disaggregate Transactions From Date parameter.
Algorithm Code	No	Used to attach a post-processing algorithm. This algorithm is triggered once the transaction legs are identified for disaggregation.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Identify Affected Transactions (C1-IAENT)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, transaction legs of the accounts for which disaggregation request is created and whose corresponding bill segments are not in the **Frozen** or **Pending Cancel** status are copied from CI_TXN_DTL_PRITM to CI_DISAGG_TXN_PRITM_DETAIL table. The corresponding aggregated billable charge ID is copied to CI_DISAGG_BCHG_DETAIL table. In addition, the system does the following:

- The **BO_STATUS_CD** column corresponding to the transaction leg in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **P**.
- The **TXN_ACT_CD** column corresponding to the transaction leg in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **DELETE (10)**.
- If non-aggregated billable charge exists for a transaction leg, the **BILLABLE_CHG_ACT_CD** column corresponding to the transaction leg in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **DELETE (10)**, **CANCEL (20)**, or **No Action on the Billable Charge (30)**. If the corresponding bill segment is in the **Freezable** status or if the bill segment is not yet generated, this column is set to **DELETE (10)**. It indicates that the non-aggregated billable charge must be deleted while executing the **Process Non Aggregated Transactions (C1-PDTXN)** batch. If the corresponding bill segment is in the **Canceled** status, this column is set to **CANCEL (20)**. It indicates that the non-aggregated billable charge must be cancelled while executing the **Process Non Aggregated Transactions (C1-PDTXN)** batch. If the corresponding billable charge is in the **Canceled** status, this column is set to **No Action on the Billable Charge (30)**. It indicates that no action must be taken on the non-aggregated billable charge.
- If aggregated billable charge exists for a transaction leg, the **BILLABLE_CHG_ACT_CD** column corresponding to the billable charge in the CI_DISAGG_BCHG_DETAIL table is set to **DELETE (10)**, **CANCEL (20)**, or **No Action on the Billable Charge (30)**. If the corresponding bill segment is in the **Freezable** status or if the bill segment is not yet generated, this column is set to **DELETE (10)**. It indicates that the aggregated billable charge must be deleted while executing the **Clean Up (C1-TXNCU)** batch. If the corresponding bill segment is in the **Canceled** status, this column is set to **CANCEL (20)**. It indicates that the aggregated billable charge must be cancelled while executing the **Clean Up (C1-TXNCU)** batch. If the corresponding billable charge is in the **Canceled** status, this column is set to **No Action on the Billable Charge (30)**. It indicates that no action must be taken on the aggregated billable charge.

3.15 Process Non Aggregated Transactions (C1-PDTXN)

The **Process Non Aggregated Transactions (C1-PDTXN)** batch is used to process the identified transactions, delete the required transaction legs, and change the status of the transaction to **Uploaded (UPLD)**. If a non-aggregated billable charge exists for a transaction leg and the corresponding bill segment is in the **Canceled** status, then:

- The billable charge is cancelled.
- The corresponding transaction leg and calculation lines (if any) are deleted.
- The status of the transaction is changed to **Uploaded (UPLD)**.

However, if a non-aggregated billable charge exists for a transaction leg, but the bill segment is not yet generated, then the billable charge, the corresponding calculation lines (if any), and transaction leg are deleted, and the status of the transaction is changed to **Uploaded (UPLD)**. If a non-aggregated billable charge is in the **Canceled** status, then the corresponding transaction leg and calculation lines (if any) are deleted and the status of the transaction is changed to **Uploaded (UPLD)**. If the rate is determined for a transaction leg which is in the **Ignored (IGNR)** status, then the calculation lines are deleted along with the transaction leg during disaggregation.

Note:

If you want to undertake some pre-processing activities (such as cleaning data in any custom tables) during the disaggregation process, you need to attach a pre-processing algorithm to the **Feed Management- Disaggregation Pre-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered when you execute the **Process Non Aggregated Transactions (C1-PDTXN)** batch. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs.

A sample pre-processing algorithm type named **C1_DSAG_PRPC** is shipped with the product. It does not have any business logic. If you want to undertake some pre-processing activities during the disaggregation process, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management- Disaggregation Pre-Processing** algorithm spot of the respective division. You can refer to the **C1_DSAG_PRPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

This batch is a multi-threaded batch. The multi-threading is based on distinct transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The records are retrieved from the **CI_DISAGG_TXN_DTL_PRITM** table. You can specify either of the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Account ID	No	Used when you want to disaggregate transactions of a particular account.
Division	No	Used when you want to disaggregate transactions of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to disaggregate transactions of accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	Used when you want to disaggregate transactions which were performed from a particular date onwards. You must specify the date in the YYYY-MM-DD format.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. <div>Note: By default, the thread count is set to 0.</div>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.

Parameter Name	Mandatory (Yes or No)	Description
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Process Non Aggregated Transactions (C1-PDTXN)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the corresponding data is deleted from the CI_TXN_DTL_PRITM, CI_TXN_DETAIL_EXCP, and CI_ROLLBACK_TXN_DETAIL tables. If the transaction in the CI_TXN_DETAIL table is disaggregated, then:

- The affected transaction leg is deleted from the CI_TXN_DTL_PRITM table.
- The non affected transaction leg is moved to the CI_TXN_DTL_PRITM_STG table.
- The transaction is moved from the CI_TXN_DETAIL to CI_TXN_DETAIL_STG table and the status of the transaction is changed to **Uploaded (UPLD)**.

When a partially disaggregated transaction in the CI_TXN_DETAIL_STG table is disaggregated, then the affected transaction leg is deleted from the CI_TXN_DTL_PRITM_STG table.

If a transaction leg is in the **COMPLETE (40)** or **IGNORED (20)** status, the **IS_DISGG** column corresponding to the transaction leg in CI_TXN_DTL_PRITM_STG table is set to **N**. Otherwise, it is set to **Y**. **N** indicates that the transaction leg should not be considered during the aggregation process whereas **Y** indicates that the transaction leg should be considered during the aggregation process.

If the **BILLABLE_CHG_ACT_CD** column corresponding to the transaction legs in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **DELETE (10)**, the non-aggregated billable charge records are deleted from the CI_BILL_CHG_K, CI_BILL_CHG, CI_BCHG_SQ, CI_BILL_CHG_CHAR, CI_B_CHG_LINE, and CI_B_LN_CHAR tables.

If the **BILLABLE_CHG_ACT_CD** column corresponding to the transaction legs in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **CANCEL (20)**, the status of the non-aggregated billable charges is changed to **Canceled** in the CI_BILL_CHG table. Finally, the **BO_STATUS_CD** column corresponding to the transaction leg in the CI_DISAGG_TXN_PRITM_DETAIL table is set to **C**.

Once a transaction leg is deleted, the corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables when effective pricing of the transaction leg has either of the following set of attributes:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **No**, and **Rating Criteria** is set to **Rate Transactions (RITX)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**

3.16 Update Disaggregation Request Status (C1-DARSU)

The **Update Disaggregation Request Status (C1-DARSU)** batch is used to change the status of the disaggregation request in the CI_TXN_DISAGG_REQ table to COMPLETE.

This batch is a multi-threaded batch. The multi-threading is based on disaggregation request ID and chunks for multi-threading are created based on numerical distribution of disaggregation request ID. The disaggregation requests which are in the **Pending** status are retrieved from the CI_TXN_DISAGG_REQ table. You can specify either of the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Account ID	No	Used when you want to update disaggregate request status of a particular account.
Division	No	Used when you want to update disaggregate request status of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to update disaggregate request status of accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	<p>Used when you want to update disaggregation request status of accounts that satisfy the following conditions:</p> <ul style="list-style-type: none"> Transactions for that account were performed from a particular date onwards. Bill segments created for the transactions are in the Pending Cancel or Frozen status. <p>You must specify the date in the YYYY-MM-DD format.</p>
Thread Count	No	<p>Used to specify the number of threads you want to spawn in parallel.</p> <div style="border: 1px solid black; padding: 2px;"> <p>Note: By default, the thread count is set to 0.</p> </div>
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Update Disaggregation Request Status (C1-DARSU)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the BO_STATUS_CD column corresponding to the disaggregation requests (which are successfully executed) is set to COMPLETE in the CI_TXN_DISAGG_REQ table.

3.17 Pending Bill Deletion (C1-DELBL)

The **Pending Bill Deletion (C1-DELBL)** batch is used to delete the bills (with the **Pending** status) and their corresponding bill segments. A pending bill and its bill segments are deleted only when the bill segments in the **Frozen** or **Pending Cancel** status are not present in the bill. If a pending bill contains a bill segment in the **Canceled** status, all other bill segments in the bill are deleted, but the bill segment in the **Canceled** status and the bill are not deleted.

This batch is used during the cancellation process. This batch is a multi-threaded batch. The multi-threading is based on bill ID and chunks for multi-threading are created based on numerical distribution of bill ID. The records are retrieved from the CI_BILL table. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	Yes	Used when you want to delete bills which include charges for transactions received through a particular feed.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Pending Bill Deletion (C1-DELBL)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the bill records are deleted from the CI_BILL, CI_BILL_K, CI_BILL_CHAR, CI_BILL_MSGS, CI_BILL_EXCP, CI_BILL_MSG_PRM, CI_BILL_ROUTING, and CI_BILL_SA tables. The corresponding bill segment details are deleted from the CI_BSEG, CI_BSEG_K, CI_BSEG_ITEM, CI_BSEG_SQ, CI_BSEG_READ, CI_BSEG_MSG, CI_BSEG_EXT, CI_BSEG_EXCP, CI_BSEG_CL_CHAR, CI_BSEG_CALC, and CI_BSEG_CALC_LN tables. The details of financial transactions (FTs) created corresponding to the bill segments are also deleted from the CI_FT, CI_FT_GL, CI_FT_K, CI_FT_PROC, and CI_FT_GL_EXT tables.

3.18 Cancellation (C1-TXCNC)

The **Cancellation (C1-TXCNC)** batch is used to delete transaction legs during the cancellation process. Finally, this batch changes the status of the feed and all transactions in the feed to **Cancelled (CNCL)**.

Note:

If you want to undertake some pre-processing activities (such as cleaning data in any custom tables) during the cancellation process, you need to attach a pre-processing algorithm to the **Feed Management Cancellation Pre-Processing** algorithm spot in the **Algorithms** tab of the **Division** screen. This algorithm is triggered when you execute the **Cancellation (C1-TXCNC)** batch. Note that the system invokes the algorithm which is attached on the derived account's division and not on the division to which the transaction belongs.

A sample pre-processing algorithm type named **C1_CNCL_PRPC** is shipped with the product. It does not have any business logic. If you want to undertake some pre-processing activities during the cancellation process, you need to create custom algorithm type and attach the respective algorithm to the **Feed Management Cancellation Pre-Processing** algorithm spot of the respective division. You can refer to the **C1_CNCL_PRPC** algorithm type to understand the input parameters that must be passed in the custom algorithm type.

This batch is a multi-threaded batch. The multi-threading is based on transaction ID and chunks for multi-threading are created based on numerical distribution of transaction ID. The records are retrieved from the CI_TXN_DETAIL and CI_TXN_DETAIL_STG tables. You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	Yes	Used when you want to cancel a particular transaction feed.
Thread Count	No	Used to specify the number of threads you want to spawn in parallel. Note: By default, the thread count is set to 0.
Chunk Size	Yes	Used to specify the number of transactions you want to execute in each work unit.
Maximum Batch Count	Yes	Used to specify the maximum number of transactions after which the data must be transferred to the database.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Cancellation (C1-TXCNC)** batch fails or aborts due to some reason, you can restart the batch over and over again with the same set of parameters.

Post Execution Check/Clean Up:

On successful completion of this batch, the status of the feed is changed to **Cancelled (CNCL)** in the CI_TXN_HEADER table. The status of all transactions in the feed is changed to **Cancelled (CNCL)** in the CI_TXN_DETAIL table. The corresponding transaction legs are deleted from the CI_TXN_DTL_PRITM table. If some transactions in the feed are in the **Error (EROR)** status, then the corresponding records are deleted from the CI_TXN_DETAIL_EXCP and CI_ROLLBACK_TXN_DETAIL tables. If some transactions in the feed are in the **Ignored (IGNR)** status, then the corresponding records are deleted from the CI_ROLLBACK_TXN_DETAIL table. If some of the transactions in the feed are partially disaggregated, then these partially disaggregated transactions are moved from the CI_TXN_DETAIL_STG to CI_TXN_DETAIL table and their status is changed to **Cancelled (CNCL)**. The corresponding transaction legs are deleted from the CI_TXN_DTL_PRITM_STG table.

If a feed is cancelled before executing the **Transaction Validation and Initial Product Determination (C1-TXNIP)** batch, all transactions in the feed are moved from the CI_TXN_DETAIL_STG to CI_TXN_DETAIL table and their status is changed to **Cancelled (CNCL)**.

When the transaction legs deleted from the CI_TXN_DTL_PRITM and CI_TXN_DTL_PRITM_STG tables have either of the following set of pricing attributes, the corresponding calculation records are deleted from the CI_TXN_CAL, CI_TXN_CALC_LN, CI_TXN_CALC_LN_CHAR, and CI_TXN_SQ tables:

- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **No**, and **Rating Criteria** is set to **Rate Transactions (RITX)**
- **Ignore Transaction** is set to **No**, **Aggregate Transaction** is set to **Yes**, and **Rating Criteria** is set to **Rate individual transactions and aggregate calc lines across transactions (RITA)**

Note:

The system will not cancel the feed when:

>> A pending bill (which is generated for the feed that you want to cancel) has a bill segment in the **Frozen** or **Pending Cancel** status.

>> A bill in the **Complete** status already exists for the feed that you want to cancel.

4. Recommended Parameter Values

This section recommends parameter values for each batch. The actual values to achieve maximum performance will vary with different hardware set. The recommendations are based on the number of CPUs and RAM available on the database and application server. The actual performance would depend on the number of CPUs and RAM available on the application server, and many other hardware parameters. Oracle Revenue Management and Billing provides various parameters which can be used for tuning batch performance as per the available hardware.

The following recommendations must be treated as guidelines and not as the actual values:

Batch Name	Batch Parameter	Recommended Value
C1-TXNRB	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-TXNRP	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNIP	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-PRMGC	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Commit Size	5000 Transactions per 16 GB of RAM
C1-TXNVP	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNEX	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-TXNSQ	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-TXNCM	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-TXNCU	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM

Batch Name	Batch Parameter	Recommended Value
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-IAENT	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-PDTXN	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-DARSU	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXCNC	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-DELBL	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
	Maximum Batch Count	5000 Transactions per 16 GB of RAM