Oracle® Revenue Management and Billing

Version 2.3.0.2.0

Transaction Feed Management (TFM) - Batch Execution Guide

Revision 4.2

E58973-01

November, 2014

ORACLE

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

E58973-01

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
Oracle Revenue Management and Billing Banking User Guide	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.

Change Log

Revision	Last Update	Updated Section	Comments
4.1	20-Jan-2015	Section 4: Recommended Parameter Values	Added Information
4.2	27-Oct-2015	Section 3.1: Rollback (C1-TXNRB)	Added Note
		Section 3.2: Flush All Caches (F1-Flush)	Added Note
		Section 3.3: Refresh Pricing (C1-TXNRP)	Added Note
		Section 3.4: Header Validation (C1-TXNHV)	Added Note
		Section 3.5: Transaction Validation and Initial Product Determination (C1-TXNIP)	Added Note
		Section 3.6: Product Pricing Verification (C1-TXNVP)	Added Note
		Section 3.7: Update Status (C1-TXNEX)	Added Note
		Section 3.8: Service Quantity Calculation (C1-TXNSQ)	Added Note
		Section 3.9: Mark Completion (C1-TXNCM)	Added Note
		Section 3.10: Clean Up (C1-TXNCU)	Added Note
		Section 3.11: Disaggregation Request Creation (C1-DISTG)	Added Note
		Section 3.12: Identify Affected Transactions (C1-IAENT)	Added Note
		Section 3.13: Process Non Aggregated Transactions (C1-PDTXN)	Added Note
		Section 3.14: Update Disaggregation Request Status (C1-DARSU)	Added Note
		Section 3.15: Pending Bill Deletion (C1-DELBL)	Added Note
		Section 3.16: Cancellation (C1-TXCNC)	Added Note

Contents

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
- Create and Update Billable Charge with the SQI values
- Clean-up Unwanted Data

The transaction feed management process includes various sub-processes, such as Header Validation, Transaction Validation and Initial Product Determination, Product Pricing Verification, Aggregation, Clean Up, Disaggregation, Cancellation, and 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 tasks in TFM:

TFM Task	Batch Sequence
Aggregation	Execute the following batches in the specified order: 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**) **Note:** 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. 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.
Rollback Transactions with the **Error (EROR)** or **Ignored (IGNR)** Status	Execute the Rollback (C1-TXNRB) batch.
Disaggregation	Execute the following batches in the specified order: 1. Disaggregation Request Creation (C1-DISTG) 2. Identify Affected Transactions (C1-IAENT) 3. Process Non Aggregated Transactions (C1-PDTXN) 4. Clean Up (C1-TXNCU) (with the **Request Type** parameter set to **DISAGG**) 5. Update Disaggregation Request Status (C1-DARSU)

TFM Task	Batch Sequence
Cancellation	Execute the following batches in the specified order: 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, its status is changed to **Uploaded (UPLD)**.

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 were ignored or where an error occurred. The valid values are: • IGNR • EROR
Transaction Header ID	No	Used when you want to rollback transactions of 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.
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.
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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. 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.

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 cache the product pricing information, and thereby improve the **Product Pricing Verification (C1-TXNVP)** batch performance. If you set the **Materialized View Used** option type of the **C1_FM** feature configuration to **true**, the system will cache the regular and post processing product pricing information in the following tables:

- CI_PRC_AGRD
- CI_PRC_PL
- CI_PRC_INH_PL

But, if you set the **Materialized View Used** option type of the **C1_FM** feature configuration to **false**, the system will not cache the regular and post processing product pricing information. If there are any pricing changes, you will have to update this cache. The **Refresh Pricing (C1-TXNRP)** batch is used to update the regular and post processing product pricing information in the cache.

You can specify the following parameters while executing this batch:

Parameter Name	Mandatory (Yes or No)	Description
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
Chunk Size	Yes	Used to specify the number of persons whose regular and post processing product pricing information you want to cache 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 cache. During this process, the existing data is first removed from the CI_PRC_AGRD, CI_PRC_PL, and CI_PRC_INH_PL tables, and then the latest regular and post processing product pricing 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.

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	Used to indicate whether the following should be validated: • 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. The valid values are: • Y • N
Duplicate Check Required	No	Used to indicate whether the following should be validated: • File with the same header date and external header ID is not available in the system. The valid values are: • Y • N **Note:** 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. 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.
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.

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 rules and product parameters which are effective on the batch business date. If the **Processing Date for Transaction Feed Management** characteristic type is set to **BATCH_DT** for the division, the system executes the rules which are effective on the batch business date. However, if the **Processing Date for Transaction Feed Management** characteristic type is set to **TXN_DT** for the division, the system executes the rules which are effective on the transaction date. But, note that the transaction date must be earlier 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. The batch business date specified while executing the C1-TXNIP batch is stamped in the database. It is also used while executing all other consequent batches in the transaction aggregation cycle.

Parameter Name	Mandatory (Yes or No)	Description
Transaction Header ID	No	Used when you want to validate and derive product for transactions in 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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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)**.
- 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)** or **Error (EROR)**.
- 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.

- A record is added for each transaction leg in the CI_TXN_DTL_PRITM_SUMMARY table. 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.

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 **Disaggregated** flag is set for each transaction leg. For newly created transaction legs, the **Disaggregated** flag is set to **Y**. And, for all existing transaction legs, the **Disaggregated** flag 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.
- A record is added for newly created transaction leg in the CI_TXN_DTL_PRITM_SUMMARY table. The SUMMARY ID is generated automatically for the 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.

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

The following table lists the statements that you must execute to gather statistics:

Table Name	Statement

CI_TXN_DETAIL	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);
	END;

CI_TXN_DTL_PRITM	BEGIN
	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);
	END;

CI_ROLLBACK_TXN_DETAIL	BEGIN
	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);
	END;

CI_PRICEITEM_PARM_GRP_K	BEGIN
	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);
	END;

CI_TXN_DTL_PRITM_SUMMARY	BEGIN
	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);
	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

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 in 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.
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
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. If the multi parameter based pricing feature is enabled, the summary record contains pricing and aggregation information for account, division, product, and group ID combination on the processing date. If the multi parameter based pricing feature is disabled, the summary record contains pricing and aggregation information for account, division, product, and TOU combination on the processing date.

3.7 Update Status (C1-TXNEX)

The **Update Status (C1-TXNEX)** batch is used to update the status of the transaction. If a transaction is ignored and not considered for billing for all products (to which it is mapped), the status of the transaction is changed to **Ignored (IGNR)**. However, if the effective pricing is not available for one or more products to which a transaction is mapped, the status of the transaction is changed to **Error (EROR)**.

In addition, the status of the transaction 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.

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 in 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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 ignored and 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 ignored and not considered for billing for one or more products and 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. 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.
- 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.
- 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.

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. 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.
- 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.
- 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.

3.8 Service Quantity Calculation (C1-TXNSQ)

The **Service Quantity Calculation (C1-TXNSQ)** batch is used to aggregate the transactions, create billable charges, and then update the SQI values in the billable charges.

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 in a particular feed.
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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 are added in the CI_BCHG_SQ table.

If the **Aggregate Transaction** field is set to **Yes**, the billable charge ID is updated in the CI_TXN_DTL_PRITM_SUMMARY table corresponding to the transaction leg. However, if the **Aggregate Transaction** field is set to **No**, the billable charge ID is updated in the CI_TXN_DTL_PRITM table corresponding to the transaction leg. 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, the status of the record is changed to **E**.

3.9 Mark Completion (C1-TXNCM)

The **Mark Completion (C1-TXNCM)** batch is used to update the status of the transaction. If the SQI values are updated successfully in the billable charge, the status of the transaction 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 if the exchange rate is not available during currency conversion, the status of the transaction is changed to **Error (EROR)**.

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 in 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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 table, respectively.

If a 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 table, 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 is changed to **Error (EROR)**. In addition, the status of the corresponding transaction leg is changed to **Error (EROR)**.

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.

3.10 Clean Up (C1-TXNCU)

The **Clean Up (C1-TXNCU)** batch is used to delete non aggregated billable charges, if any, created for transactions in the **Error (EROR)** status. It also recalculates SQIs in a billable charge if the transactions in the **Error (EROR)** and **Completed (COMP)** status are aggregated together in the billable charge.

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

- **Cancellation** - During the cancellation process, it deletes non aggregated billable charges and recalculates SQIs in aggregated billable charges.
- **Disaggregation** - During the disaggregation process, it deletes an aggregated billable charge when all the corresponding transaction legs which were aggregated in the billable charge are deleted during disaggregation.

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.

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 in a particular feed. **Note:** This parameter should not be used during the disaggregation process. This parameter is required when you set the request type to **CNCL**.
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.

Parameter Name	Mandatory (Yes or No)	Description
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: • 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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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, and CI_BCHG_SQ tables. During the aggregation process, the records are deleted from the CI_TXN_DTL_PRITM_SUMMARY table.

During the cancellation process, the non aggregated billable charges are deleted from the CI_BILL_CHG, CI_BILL_CHG_K, and CI_BCHG_SQ tables. In addition, the transaction legs which are cancelled are deleted from the CI_TXN_DTL_PRITM table. If an aggregated billable charge includes legs of transactions from more than one feed, the SQIs are recalculated in the aggregated billable charge. 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, and CI_BCHG_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 charges are deleted from the CI_BILL_CHG_K, CI_BILL_CHG and CI_BCHG_SQ tables. The corresponding bill segments are deleted from the CI_BSEG, CI_BSEG_K, CI_BSEG_ITEM, CI_BSEG_SQ, CI_BSEG_READ, CI_BSEG_MSG, CI_BSEG_EXCP, CI_BSEG_CL_CHAR, CI_BSEG_CALC, and CI_BSEG_CALC_LN tables. In addition, the corresponding financial transactions (FTs) are deleted from the CI_FT, CI_FT_GL, CI_FT_K, CI_FT_PROC, and CI_FT_GL_EXT 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 **Cancelled** 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**.

3.11 Disaggregation Request Creation (C1-DISTG)

The **Disaggregation Request Creation (C1-DISTG)** batch is used to create a disaggregation request for an account whose transactions need to be 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 at 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 at A1 has not changed.

This batch is a single-threaded batch. 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 request for an account (account ID) or a customer (person ID) whose transactions need to be disaggregated is 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**.

3.12 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 and identify the transactions that need to be disaggregated.

Pre-requisites:

- A disaggregation request for an account whose transactions need to be disaggregated must be present in the CI_TXN_DISAGG_REQ table.
- Before you execute the **Identify Affected Transactions (C1-IAENT)** batch, you need to ensure that there are no pending bills for the accounts whose transactions need to be disaggregated. If there are pending bills for these accounts, you need to 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. For more information about these batches, refer to *Oracle Revenue Management and Billing Batch Execution Guide*.

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.
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
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 charges are 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

either set to **DELETE (10)** or **CANCEL (20).** 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 **Cancel** 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 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 either set to **DELETE (10)** or **CANCEL (20).** 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 **Cancel** 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.

3.13 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)**. In addition, if non aggregated billable charge exists for a transaction leg and if the corresponding bill segment is in the **Freezable** status or if the bill segment is not yet generated, it deletes the non aggregated billable charge. However, if non aggregated billable charge exists for a transaction leg and if the corresponding bill segment is in the **Cancel** status, it cancels the non aggregated billable charge.

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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 and CI_ROLLBACK_TXN_DETAIL tables. The status of the transaction is changed to **Uploaded (UPLD)** in the CI_TXN_DETAIL table. However, when a partially disaggregated transaction is disaggregated, the corresponding data is deleted from the CI_TXN_DTL_PRITM_STG table. The status of the transaction is changed to **Uploaded (UPLD)** in the CI_TXN_DETAIL_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 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 charges are deleted from the CI_BILL_CHG_K, CI_BILL_CHG and CI_BCHG_SQ tables. The corresponding bill segments are deleted from the CI_BSEG, CI_BSEG_K, CI_BSEG_ITEM, CI_BSEG_SQ, CI_BSEG_READ, CI_BSEG_MSG, CI_BSEG_EXCP, CI_BSEG_CL_CHAR, CI_BSEG_CALC, and CI_BSEG_CALC_LN tables. In addition, the corresponding financial transactions (FTs) are deleted from the CI_FT, CI_FT_GL, CI_FT_K, CI_FT_PROC, and CI_FT_GL_EXT 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 **Cancelled** 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**.

3.14 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. 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.

Parameter Name	Mandatory (Yes or No)	Description
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	Used when you want to update disaggregation request status of accounts that satisfy the following conditions: • 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 You must specify the date in the YYYY-MM-DD format.
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
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) in the CI_TXN_DISAGG_REQ table is set to COMPLETE.

3.15 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. This batch is used during the cancellation process. 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 are created for transactions in a particular transaction feed.
Thread Count	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 bills are deleted from the CI_BILL and CI_BILL_K tables. The corresponding bill segments are deleted from the CI_BSEG, CI_BSEG_K, CI_BSEG_ITEM, CI_BSEG_SQ, CI_BSEG_READ, CI_BSEG_MSG, CI_BSEG_EXCP, CI_BSEG_CL_CHAR, CI_BSEG_CALC, and CI_BSEG_CALC_LN tables. The financial transactions (FTs) created corresponding to the bills (which are deleted) are also deleted from the CI_FT, CI_FT_GL, CI_FT_K, CI_FT_PROC, and CI_FT_GL_EXT tables.

3.16 Cancellation (C1-TXCNC)

The **Cancellation (C1-TXCNC)** batch is used to change the status of the feed and all transactions in the feed to **Cancelled (CNCL)**. 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	Yes	Used to specify the number of threads you want to spawn in parallel.
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 committed in 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 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)**.

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 **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
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNRB	Maximum Batch Count	5000 Transactions per 16 GB of RAM
C1-TXNRP	Chunk Size	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNIP	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
C1-TXNVP	Chunk Size	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNEX	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNSQ	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNCM	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
	Chunk Size	5000 Transactions per 16 GB of RAM
C1-TXNCU	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
C1-IAENT	Chunk Size	5000 Transactions per 16 GB of RAM

Batch Name	Batch Parameter	Recommended Value
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