Oracle® Revenue Management and Billing

Version 5.0.0.0.0

Transaction Feed Management (TFM) - Batch Execution Guide

Revision 22.0

F75571-01

October 2023

ORACLE

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

F75571-01

Trademark Notice

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

Intel and Intel Inside 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, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

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.

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 Price Item
- Verify Price Item Pricing
- Rate Transactions Before Billing (if required)
- Create and Update Billable Charge with the SQI 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 price items 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 Price Item Determination, Price Item 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.

Note: If you change the TFM setup, such as divisions' characteristics, divisions' algorithms, SQIs defined for price item and division combinations, price item parameters, rules, and so on, you must restart the thread pool before executing any batch.

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	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. Validate Transaction and Derive Price Item (C1-TXNIP) 5. Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS) 6. Price Item Pricing Verification (C1-TXNVP) 7. Update Status (C1-TXNEX) 8. Service Quantity Calculation (C1-TXNSQ) 9. Mark Completion (C1-TXNCM) 10. 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 **Validate Transaction and Derive Price Item (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. The system allows you to execute each batch consecutively. You can execute the **Validate Transaction and Derive Price Item (C1-TXNIP)** batch consecutively with the same division and same parameters, or with the different division and different parameters. But, you cannot execute the **Validate Transaction and Derive Price Item (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. You must execute the **Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS)** batch only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to true.

TFM Sub-Process	Batch Sequence
	Once the **Price Item Pricing Verification (C1-TXNVP)** batch is executed, you cannot execute the **Validate Transaction and Derive Price Item (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 **Price Item 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. During the transaction aggregation process, you must specify the same division and same parameters across each batch. Otherwise, erroneous results might occur. If you have already executed the **Validate Transaction and Derive Price Item (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.
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. Pending Bill Segments Deletion (C1-BSEGD) 3. Pending Bill Deletion (C1-PNBD) 4. Identify Transactions for Disaggregation (C1-IDENT) 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-DRSUA)
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, 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 **TFM - Rollback Pre-Processing** 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 **TFM - Rollback Pre-Processing** 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** or **Error** status. The valid values are: • IGNR • EROR
Transaction Header ID	No	Used when you want to rollback transactions which are received in a particular transaction feed.

Parameter Name	Mandatory (Yes or No)	Description
Transaction Source	No	Used when you want to rollback transactions which are 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. **Note:** You must specify the date in the YYYY-MM-DD format. 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. **Note:** You must specify the date in the YYYY-MM-DD format. The rollback to date cannot be earlier than the rollback from date.
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**, **CI_TXN_SQ**, and **CI_TXN_CALC_ACCUM_GRP** 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 price item pricing information in the following tables and thereby improve the performance of the **Price Item 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 price item 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 price item 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 price item pricing information while executing the **Price Item Pricing Verification (C1-TXNVP)** or **Bill Completion (C1-BLPPR)** batch, respectively. The system stores all price items' 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 **Price Item 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 price item 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
Division	No	Used when you want to update the price item pricing information of accounts belonging to a particular division.
Chunk Size	Yes	Used to specify the number of persons whose regular and post-processing price item 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 price item 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 the transaction feeds which are 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:** The duplicate check is done only against the transaction feeds which are in the **Validated (VALI)** status and not against the transaction feeds which are in the **Uploaded (UPLD)** status.
Allow Positive Transaction Volume in Header	Yes	Used to indicate whether you want to allow positive value in the total transaction volume which is specified in the header record. The valid values are: • Y • N **Note:** By default, the parameter value is set to **Y**.
Allow Negative Transaction Volume in Header	Yes	Used to indicate whether you want to allow negative value in the total transaction volume which is specified in the header record. The valid values are: • Y • N **Note:** By default, the parameter value is set to **N**.

Parameter Name	Mandatory (Yes or No)	Description
Allow Zero Transaction Volume in Header	Yes	Used to indicate whether you want to allow zero value in the total transaction volume which is specified in the header record. The valid values are: • Y • N **Note:** By default, the parameter value is set to **N**.
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 Validate Transaction and Derive Price Item (C1-TXNIP)

The **Validate Transaction and Derive Price Item (C1-TXNIP)** batch is used to validate transaction level information. If the transaction validation is successful, the system checks whether a primary pricing rule type is specified in the transaction record type. If a primary pricing rule type is specified in the transaction record type, the system calls the primary pricing rule type and invokes the algorithms which are attached to the following system events of the primary pricing rule type in the specified sequence:

1. **Transaction Validation** - At present, the product has not shipped an algorithm type for the **Transaction Validation** system event. If required, you can create a custom algorithm type which validates the transaction before deriving the transaction legs.
2. **Bill Group Derivation** - For more information, refer to the **Bill Group Derivation** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*.

3. **Account and Price Item Derivation** - Depending on the category to which the primary pricing rule type belongs, you need to attach different algorithm to the **Account and Price Item Derivation** system event. The following table lists the algorithm that you can attach to the primary pricing rule types of different category:

Pricing Rule Type Category	Algorithm	For more information on how the system derives the transaction legs, refer to...
Claim	C1_ACCPRIDRV	**Account and Price Item Derivation (for the Claim Pricing Rule Type Category)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*
Retention Type Enrollment Based	C1_ACCPRIDRV	**Account and Price Item Derivation (for the Retention Type Enrollment Based Pricing Rule Type Category)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*
Ancillary	C1_ACCPRIDRV	**Account and Price Item Derivation (for the Ancillary Pricing Rule Type Category)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*

Once the primary pricing rule type is called, the system calls the eligible related pricing rule types (if any) defined in the primary pricing rule type. Note that the related pricing rule types are called one by one in the specified sequence. The system invokes the algorithms which are attached to the following system events of the related pricing rule type in the specified sequence:

1. **Transaction Validation** - At present, the product has not shipped an algorithm type for the **Transaction Validation** system event. If required, you can create a custom algorithm type which validates the transaction before deriving the transaction legs.
2. **Account and Price Item Derivation** - Depending on the category to which the related pricing rule type belongs, you need to attach different algorithm to the **Account and Price Item Derivation** system event. The following table lists the algorithm that you can attach to the related pricing rule types of different category:

Pricing Rule Type Category	Algorithm	For more information on how the system derives the transaction legs, refer to...
Specific Stop-Loss	C1_ACCPRISL	**Account and Price Item Derivation (for the Specific Stop-Loss and Aggregate Stop-Loss Pricing Rule Type Categories)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*

Pricing Rule Type Category	Algorithm	For more information on how the system derives the transaction legs, refer to...
Aggregate Stop-Loss	C1_ACCPRISL	**Account and Price Item Derivation (for the Specific Stop-Loss and Aggregate Stop-Loss Pricing Rule Type Categories)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*
Retention Type Claim Based	C1_ACCPRIDRV	**Account and Price Item Derivation (for the Retention Type Claim Based Pricing Rule Type Category)** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*

However, if a primary pricing rule type is not specified in the transaction record type, the system determines the account (which will bear the charges for the transaction) and the price item and variance parameter or price item parameters (to which the transaction will be mapped) using the business rules which are invoked through the rule type. Depending on the transaction record type, a rule type is invoked for each transaction. The effective rules created using the specified rule type are executed, starting with the rule having highest priority, until a rule is satisfied. For example, a rule with the priority 10 will be executed before a rule with the priority 20.

Once a rule is satisfied, the transaction is mapped to either of the following:

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

A transaction leg is created for each account, price item, and variance parameter or price item parameters combination. Then, the price item parameters to which the transaction leg is mapped are grouped. A set of price item parameters are grouped only when the multi parameter based pricing feature is enabled. This group is then used to determine the price item pricing.

Note for the Financial Services Domain:

Based on the **TFM - Processing Date** characteristic defined for the division to which a transaction belongs, the system determines the rules, price item parameters, pricing, and exchange rate which are effective on the transaction or batch business date. The effective price item parameters, pricing, and exchange rate are determined for all account and price item combinations to which the transaction is mapped. If you want to use a custom processing date for a particular account and price item 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 price item combination. However, the overridden processing date is used only while determining effective price item 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 price item 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 for the Healthcare Domain:

The derivation date is used as the processing date while executing the **C1-TXNIP** batch. The processing date is stamped in the database. It is then used while executing all subsequent batches in the transaction aggregation cycle.

Note:

At present, the transaction legs are aggregated in a billable charge based on the account ID, price item, variance or price item parameters, aggregation start date, and aggregation end date. In the Financial Services domain, the price item parameters for which the parameter usage is set to **Pricing** or **Aggregation** are used while aggregating the transaction legs. However, in the Healthcare domain, the price item parameters for which the parameter usage is set to **Aggregation** are only used while aggregating the transaction legs.

You can also aggregate transaction legs in a billable charge using additional aggregation parameters. The additional aggregation parameters can be stamped in the aggregation parameter group 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 **TFM - Price Item 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 Price Item Determined (INPD)** status. It is triggered before the price item 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 **TFM - Price Item 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 price item. The system considers the transactions whose transaction date is 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 used while executing the **C1-TXNIP** batch is stamped in the database. It is then used while executing all other consequent batches in the transaction aggregation cycle.
Transaction Header ID	No	Used when you want to validate and derive price item for the transactions which are received in a particular transaction feed.
Transaction Source	No	Used when you want to validate and derive price item for the transactions which are received from a particular transaction source.
Division	No	Used when you want to validate and derive price item for the transactions belonging to a particular division.
Shuffle Work Unit	No	Used to indicate whether you want to shuffle the work units across threads to correct the uneven thread processing time. The valid values are: • Y • N **Note:** By default, the parameter value is set to **N**.
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 **Validate Transaction and Derive Price Item (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 Price Item 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 Price Item 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 Price Item 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 ID is generated for each price item parameter group and added in the **CI_PRICEITEM_PARM_GRP_K** table. In addition, the price item parameter group ID is added in the **CI_TXN_DTL_PRITM** table against the corresponding transaction legs.
- If a price item parameter group with a set of price item parameters already exists in the system, a new price item parameter group is not created. Instead, the existing price item parameter group is used for determining the price item pricing. If the price item parameters are not derived along with the price item, the price item parameter 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 price item 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.
- In the Financial Services domain, if a transaction leg is aggregated, the aggregation parameters are grouped and the aggregation parameter group ID is generated. The aggregation parameter group includes the account ID, price item, price item parameters (where the parameter usage is set to **Pricing** or **Aggregation**), aggregation start date, and aggregation end date. The aggregation parameter group ID is added in the **AGG_PARM_GRP_ID** column corresponding to the transaction leg in the **CI_TXN_DTL_PRITM** table. However, if a transaction leg is not aggregated, the aggregation parameter group ID corresponding to the transaction leg is set to **1**. If the aggregation parameter is mandatory for the price item and if it is not given in the transaction, 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 the Healthcare domain, if a transaction leg is aggregated, the aggregation parameters are grouped and the aggregation parameter group ID is generated when the following conditions are met:
    - Aggregation parameters are added in the pricing rule type
    - Aggregation parameters are effective for the price item on the derivation date

- Claim, enrollment, or ancillary transaction contains the aggregation information

The aggregation parameter group includes the account ID, price item, price item parameters (where the parameter usage is set to **Aggregation**), aggregation start date, and aggregation end date. The aggregation parameter group ID is added in the **AGG_PARM_GRP_ID** column corresponding to the transaction leg in the **CI_TXN_DTL_PRITM** table. However, if a transaction leg is not aggregated or if any of the above condition is not met, the aggregation parameter group ID corresponding to the transaction leg is set to **1**. However, if the aggregation parameter is mandatory for the price item and if it is not given in the transaction, 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.

- One record is added for a set of transaction legs (which are in the **Initial Price Item Determined (INPD)** status and which have the same account, price item, variance parameter or price item 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.

Note: The **Validate Transaction and Derive Price Item (C1-TXNIP)** batch adds records in the **CI_TXN_DTL_PRITM_SUMMARY** table only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **false**.

- A hash value is computed using the oracle function named **ORA_HASH()** and then added in the **SUMM_HASHCODE** column corresponding to the transaction leg in the **CI_TXN_DTL_PRITM** table. This function uses the following columns of the **CI_TXN_DTL_PRITM** table to compute the hash value for each transaction leg:
  - ACCT_ID
  - INITIAL_PRICE_ITEM_CD
  - INTIAL_TOU_CD
  - PRICEITEM_PARM_GRP_ID
  - AGG_PARM_GRP_ID
  - TXN_DTTM
  - PROCESSING_DT

Note: The hash value is added in the **SUMM_HASHCODE** column corresponding to each transaction leg in the **CI_TXN_DTL_PRITM** table only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **true**.

- 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 re-aggregated 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 Price Item 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 Price Item 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 ID is generated for each price item parameter group. Note that the price item parameter group ID is generated only for the transaction legs which are newly created. The price item parameter 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 price item parameter group with a set of price item parameters already exists in the system, a new price item parameter group is not created. Instead, the existing price item parameter group is used for determining the price item pricing. If the price item parameters are not derived along with the price item, the price item parameter group ID is set to 1 in the **CI_TXN_DTL_PRITM** table against the corresponding transaction legs which are newly created.
- If the system could not successfully create a price item parameter group ID for newly derived price item 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.
- In the Financial Services domain, if a newly created transaction leg is aggregated, the aggregation parameters are grouped and the aggregation parameter group ID is generated. The aggregation parameter group includes the account ID, price item, price item parameters (where the parameter usage is set to **Pricing** or **Aggregation**), aggregation start date, and aggregation end date. The aggregation parameter group ID is added in the **AGG_PARM_GRP_ID** column corresponding to the transaction leg in the **CI_TXN_DTL_PRITM** table. However, if a newly created transaction leg is not aggregated, the aggregation parameter group ID corresponding to the transaction leg is set to **1**. If the aggregation parameter is mandatory for the price item and if it is not given in the transaction, 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 the Healthcare domain, if a newly created transaction leg is aggregated, the aggregation parameters are grouped and the aggregation parameter group ID is generated when the following conditions are met:
    - Aggregation parameters are added in the pricing rule type
    - Aggregation parameters are effective for the price item on the derivation date
    - Claim, enrollment, or ancillary transaction contains the aggregation information

The aggregation parameter group includes the account ID, price item, price item parameters (where the parameter usage is set to **Aggregation**), aggregation start date, and aggregation end date. The aggregation parameter group ID is added in the **AGG_PARM_GRP_ID** column corresponding to the transaction leg in the **CI_TXN_DTL_PRITM** table. However, if a newly created transaction leg is not aggregated or if any of the above condition is not met, the aggregation parameter group ID corresponding to the transaction leg is set to **1**. However, if the aggregation parameter is mandatory for the price item and if it is not given in the transaction, 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.

- One record is added for a set of newly created transaction legs (which are in the **Initial Price Item Determined (INPD)** status and which have the same account, price item, variance parameter or price item 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.

Note: The **Validate Transaction and Derive Price Item (C1-TXNIP)** batch adds records in the **CI_TXN_DTL_PRITM_SUMMARY** table only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **false**.

- A hash value is computed using the oracle function named **ORA_HASH()** and then added in the **SUMM_HASHCODE** column corresponding to the newly created transaction leg in the **CI_TXN_DTL_PRITM** table. This function uses the following columns of the **CI_TXN_DTL_PRITM** table to compute the hash value for each transaction leg:
    - ACCT_ID
    - INITIAL_PRICE_ITEM_CD
    - INTIAL_TOU_CD
    - PRICEITEM_PARM_GRP_ID
    - AGG_PARM_GRP_ID
    - TXN_DTTM
    - PROCESSING_DT

Note: The hash value is added in the **SUMM_HASHCODE** column corresponding to newly created transaction leg in the **CI_TXN_DTL_PRITM** table only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **true**.

- 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

Note: You must generate statistics for the **CI_TXN_DTL_PRITM_SUMMARY** table after executing the **Validate Transaction and Derive Price Item (C1-TXNIP)** batch only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **false**.

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 Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS)

The **Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS)** batch is used to add one record for a set of transaction legs (which are in the **Initial Price Item Determined (INPD)** status and which have the same account, price item, variance parameter or price item 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.

Note: You must execute the **Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS)** batch only when the **Use C1-TXNPS During Transaction Aggregation** option type in the **C1_FM** feature configuration is set to **true**.

This batch is a multi-threaded batch. The multi-threading is based on summary hash code and chunks for multi-threading are created based on numerical distribution of summary hash code. 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 add summary records for the transactions which are received in a particular transaction feed.
Transaction Source	No	Used when you want to add summary records for the transactions which are received from a particular transaction source.
Division	No	Used when you want to add summary records for the transactions belonging to a particular division.
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 **Populate CI_TXN_DTL_PRITM_SUMMARY Table (C1-TXNPS)** 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, a record is added for a set of transaction legs 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.

Once you execute this batch, we recommend you to generate complete statistics for the **CI_TXN_DTL_PRITM_SUMMARY** table using the following statement:

```
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;
```

3.7 Price Item Pricing Verification (C1-TXNVP)

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

- An account, price item, and/or TOU combination on the processing date if the multi parameter based pricing feature is disabled

- An account, price item, and/or price item parameters 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 **TFM - 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.

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.

Note for the Healthcare Domain:

Once the effective pricing is determined, the system invokes the algorithms attached to the following algorithm spots of the derived account's division in the specified sequence:

1. **TFM - Contract Derivation** - You can attach an algorithm created using the **SA_DERV_POPC** algorithm type to this algorithm spot. If the account has multiple active contracts of the contract type which is associated with the price item, this algorithm derives the contract which is associated with the policy and maps it to the transaction leg. The manner in which the system derives the active contract for the account differs in the following scenarios:

If the effective pricing rule stamped against the transaction leg is defined at...	Then...
The bill group level	The system first fetches the policy derived for the transaction and then derives the active contract which is associated with the policy.
The parent customer level	The system first fetches the bill group to which the account belongs and then the policy where the bill group is associated with the policy using the policy person role which is specified in the **Bill Group Policy Person Role** option type of the **C1-ASOBLLNG** feature configuration. Once the policy is derived, the system derives the active contract which is associated with the policy.

2. **TFM - Verify Pricing Post-Processing** - You can attach an algorithm created using the **C1-VRPR_POPC** algorithm type to this algorithm spot. This algorithm removes the price assignment ID and price item parameter group ID from the summary record and adds the price item code to the summary record for each transaction leg.

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 find the price item pricing for transactions which are received in a particular transaction feed.
Transaction Source	No	Used when you want to find the price item pricing for transactions which are received from a particular transaction source.
Division	No	Used when you want to find the price item pricing for transactions belonging to a particular division.
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 **Price Item 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, price item, price item 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, price item, variance parameter, transaction date, processing date, and aggregation parameter group ID combination.

In the Healthcare domain, the price assignment ID and price item parameter group ID are removed from the summary record and the price item code is added to the summary record corresponding to each transaction leg.

3.8 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 Price Item 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 price items 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 **TFM - 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 **TFM - 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.

While determining the rate for a transaction leg, the system executes the calculation algorithm attached to the rate component. Once the rate is determined for transaction legs, a set of rate component characteristics (where the characteristic entity is 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 **TFM - 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 Price Item 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 which are received in a particular transaction feed.
Transaction Source	No	Used when you want to change the status of transactions which are received from a particular transaction source.
Division	No	Used when you want to change the status of transactions belonging to a particular division.
Shuffle Work Unit	No	Used to indicate whether you want to shuffle the work units across threads to correct the uneven thread processing time. The valid values are: • Y • N **Note:** By default, the parameter value is set to **N**.
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 price items (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 price items, but not for all price items (to which it is mapped), the status of the transaction remains as **Initial Price Item 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 Price Item 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 price items or for all price items (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 price item, 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 price item, the status of the corresponding transaction leg remains as **Initial Price Item 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. In the Healthcare domain, the calculation lines are generated using the calculation algorithm. If the calculation lines are generated for specific stop-loss or aggregate stop-loss transaction legs, the accumulation details are stored in the **CI_TXN_CALC_ACCUM_GRP** table.
- 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 Price Item 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 price item, 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 price item, the status of the corresponding transaction leg remains as **Initial Price Item 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. In the Healthcare domain, the calculation lines are generated using the calculation algorithm. If the calculation lines are generated for specific stop-loss or aggregate stop-loss transaction legs, the accumulation details are stored in the **CI_TXN_CALC_ACCUM_GRP** table.
- 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.

3.9 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.

Note: Neither the specific stop-loss or aggregate stop-loss transaction legs are aggregated, nor the specific stop-loss or aggregate stop-loss billable charges are created through the **Service Quantity Calculation (C1-TXNSQ)** batch.

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 do some pre-processing activities before invoking the rates engine, you need to attach a pre-processing algorithm to the **TFM - 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 **TFM - 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.

If you want to perform some post-processing activities on a billable charge, you need to attach a post-processing algorithm to the **TFM - 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.

In the Healthcare domain, the system invokes the algorithm attached to the following algorithm spot of the derived account's division:

>> TFM - Billable Charge Post Processing - You can attach an algorithm created using the **C1_BCHG_POPC** algorithm type to this algorithm spot. This algorithm invokes the algorithm which is attached to the **Bill After Date Determination** system event of the primary pricing rule type.

For more information, refer to the **Bill After Date Determination** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*.

While determining the rate for a transaction leg, the system executes the calculation algorithm attached to the rate component. Once the rate is determined for transaction legs, a set of rate component characteristics (where the characteristic entity is 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 **TFM - 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 Price Item 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 the billable charges for transactions which are received in a particular transaction feed.
Transaction Source	No	Used when you want to create the billable charges for transactions which are received from a particular transaction source.
Division	No	Used when you want to create the billable charges for transactions belonging to a particular division.
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. In the Healthcare domain, the calculation lines are generated using the calculation algorithm. If the calculation lines are generated for specific stop-loss or aggregate stop-loss transaction legs, the accumulation details are stored in the **CI_TXN_CALC_ACCUM_GRP** table.

If you re-execute the **Service Quantity Calculation (C1-TXNSQ)** batch, the calculation lines and pass through billable charge lines will be updated, accordingly.

3.10 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 price item — 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.
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 Price Item 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 which are received in a particular transaction feed.
Transaction Source	No	Used when you want to change the status of transactions which are received from a particular transaction source.
Division	No	Used when you want to change the status of transactions belonging to a particular division.
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 price items to which the transaction is mapped, the status of the transaction and its legs is changed to **Completed (COMP)** in the **CI_TXN_DETAIL** and **CI_TXN_DTL_PRITM** tables, respectively. However, if any error occurs while creating or updating the billable charge for any price item (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 the **CI_TXN_CAL**, **CI_TXN_CALC_LN**, **CI_TXN_CALC_LN_CHAR**, **CI_TXN_SQ**, and **CI_TXN_CALC_ACCUM_GRP** 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.

3.11 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.
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.

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 **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 **TFM - 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.

In the Healthcare domain, the system invokes the algorithm attached to the following algorithm spot of the derived account's division:

>> TFM - Billable Charge Post Processing - You can attach an algorithm created using the **C1_BCHG_POPC** algorithm type to this algorithm spot. This algorithm invokes the algorithm which is attached to the **Bill After Date Determination** system event of the primary pricing rule type. For more information, refer to the **Bill After Date Determination** section of the **Administrative Services Only (ASO) Billing** chapter in *Oracle Revenue Management and Billing Insurance User Guide*.

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.

Note: In the Healthcare domain, you must not use the **CNCL** request type in the **Clean Up (C1-TXNCU)** batch. This is because the Transaction Cancellation process is not yet supported for the Administrative Services Only (ASO) business.

- **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 update or delete billable charges created for transactions which are received in a particular transaction 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 update or delete billable charges created for transactions which are 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 update or delete 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 update or delete billable charges created for transactions of a particular account. **Note:** This parameter should be used only during the disaggregation process.

Parameter Name	Mandatory (Yes or No)	Description
Bill Cycle	No	Used when you want to update or delete billable charges created for transactions of accounts having a particular bill cycle. **Note:** This parameter should be used only during the disaggregation process.
Disaggregate Transactions From Date	Yes (Conditional)	Used when you want to update or delete billable charges created for transactions which were performed from a particular date onwards. **Note:** You must specify the date in the YYYY-MM-DD format. This parameter should be used only during the disaggregation process. This parameter is required when you set the request type to **DISAGG**.
Request Type	Yes	Used to indicate the process during which you want to execute the batch. The valid values are: • CNCL • EROR • DISAGG
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**, **CI_TXN_SQ**, and **CI_TXN_CALC_ACCUM_GRP** 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**, **CI_TXN_SQ**, and **CI_TXN_CALC_ACCUM_GRP** tables.

3.12 Disaggregation Request Creation (C1-DISTG)

The **Disaggregation Request Creation (C1-DISTG)** batch is used to create disaggregation request for an account. When you create a disaggregation request for an account, the transactions mapped to the account are disaggregated. The 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 price item level. Let us understand this with the help of an example. The following table lists the accounts and price items to which T1 is mapped:

Transaction	Account	Price Item
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 account ID and chunks for multi-threading are created based on numerical distribution of account ID. 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.
Person ID	No	Used when you want to create disaggregation request for accounts belonging to a particular person.
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. **Note:** 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 persons 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 person in the **CI_TXN_DISAGG_REQ** table.

3.13 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
Account ID	No	Used to indicate the account whose bill segments you want to delete.
Division	No	Used when you want to delete the bill segments of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to delete the bill segments of accounts having a particular bill cycle.
Request Type	No	Used to indicate whether you want to delete the bill segments during the disaggregation process. The valid values are: • **BILLING** – Used when you want to delete the bill segments from the pending bills of accounts that meet the criteria. • **DISAGG** – Used when you want to delete the bill segments of accounts that meet the criteria and for which the disaggregation request is created in the system. **Note:** If you do not specify any value, by default, the parameter value is set to **BILLING**.
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.

Parameter Name	Mandatory (Yes or No)	Description
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.

3.14 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
Account ID	No	Used to indicate the account whose pending bills you want to delete.
Division	No	Used when you want to delete the pending bills of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to delete the pending bills of accounts having a particular bill cycle.

Parameter Name	Mandatory (Yes or No)	Description
Request Type	No	Used to indicate whether you want to delete the pending bills during the disaggregation process. The valid values are: • **BILLING** – Used when you want to delete the pending bills of accounts that meet the criteria. • **DISAGG** – Used when you want to delete the pending bills of accounts that meet the criteria and for which the disaggregation request is created. **Note:** If you do not specify any value, by default, the parameter value is set to **BILLING**.
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.

3.15 Identify Transactions for Disaggregation (C1-IDENT)

The **Identify Transactions for Disaggregation (C1-IDENT)** batch is used to fetch disaggregation requests which are created for accounts 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 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
Division	No	Used when you want to identify the transactions of accounts belonging to a particular division for disaggregation.
Account ID	No	Used when you want to identify the transactions of a particular account for disaggregation.
Bill Cycle	No	Used when you want to identify the transactions of accounts having a particular bill cycle for disaggregation.
Disaggregate Transactions From Date	No	Used when you want to identify the transactions which were performed from a particular date onwards for disaggregation. **Note:** You must specify the date in the YYYY-MM-DD format. The aggregated billable charge, which is affected, should not contain a transaction leg whose transaction date is earlier than the date specified in this parameter. Otherwise, erroneous results will occur. Therefore, ensure that you specify the appropriate value for the **Disaggregate Transactions From Date** parameter.
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.
Post-Processing Algorithm	No	Used to attach a post-processing algorithm. This algorithm is triggered once the transactions and corresponding billable charges are identified for disaggregation.
Exclude Canceled Billable Charges (Y or N)	No	Used to indicate whether you want to exclude the aggregated and non-aggregated billable charges which are in the **Canceled** status during the disaggregation process. The valid values are: • Y • N **Note:** If you do not specify any value, by default, the parameter value is set to **N**.

Parameter Name	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Note: If the **Identify Transactions for Disaggregation (C1-IDENT)** 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 the **CI_TXN_DTL_PRITM** to **CI_DISAGG_TXN_PRITM_DETAIL** table. The corresponding aggregated billable charge ID is copied to the **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 aggr**e**gated billable charge must be deleted while executing the **Clean Up (C1-TXNCU)** batch. If the corresponding bill segment is in the **Canceled** status**,** this co**l**umn is set to **CANCEL (20)**. It indicates that the aggregated billable charge must be cancelled whil**e** executing the **Clean Up (C1-TXNCU)** batch. If the corresponding billable charge is in the **Canceled** stat**us,** 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.

3.16 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 **TFM - 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 **TFM - 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
Division	No	Used when you want to disaggregate the transactions of accounts belonging to a particular division.
Account ID	No	Used when you want to disaggregate the transactions of a particular account.

Parameter Name	Mandatory (Yes or No)	Description
Bill Cycle	No	Used when you want to disaggregate the transactions of accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	Used when you want to disaggregate the transactions which were performed from a particular date onwards. **Note:** You must specify the date in the YYYY-MM-DD format.
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 **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 the **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**, **CI_TXN_SQ**, and **CI_TXN_CALC_ACCUM_GRP** 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)**

3.17 Update Disaggregation Request Status (C1-DRSUA)

The **Update Disaggregation Request Status (C1-DRSUA)** batch is used to change the status of the disaggregation request in the **CI_TXN_DISAGG_REQ** table. It allows you to specify a custom algorithm which indicates when the status of the disaggregation request in the **CI_TXN_DISAGG_REQ** table must be changed to **COMPLETE**. If the custom algorithm is specified, the system uses the custom logic and not the in-built logic for updating the status of the disaggregation requests.

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 the disaggregation requests' status of a particular account.
Division	No	Used when you want to update the disaggregation requests' status of accounts belonging to a particular division.
Bill Cycle	No	Used when you want to update the disaggregation requests' status of accounts having a particular bill cycle.
Disaggregate Transactions From Date	Yes	Used when you want to update the disaggregation requests' status of accounts whose transactions were performed from a particular date onwards and the bill segments created for these transactions are in the **Pending Cancel** or **Frozen** status. **Note:** You must specify the date in the YYYY-MM-DD format.
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
Update Status Algorithm	No	Used to attach a custom algorithm which indicates when the status of the disaggregation request in the **CI_TXN_DISAGG_REQ** table must be changed to **COMPLETE**. **Note:** If an algorithm is specified in this parameter, the system uses the custom logic and not the in-built logic for updating the status of the disaggregation requests.
Exclude Canceled Billable Charges (Y or N)	No	Used to indicate whether you want to change the status of the disaggregation request to **COMPLETE** when the canceled billable charges are excluded during the disaggregation process. The valid values are: • Y • N **Note:** If you do not specify any value, by default, the parameter value is set to **N**. You must specify the same value for this parameter while executing the **Identify Transactions for Disaggregation (C1-IDENT)** and **Update Disaggregation Request Status (C1-DRSUA)** batches during the disaggregation process. Otherwise, erroneous results will occur.
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-DRSUA)** 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 in the **CI_TXN_DISAGG_REQ** table is set to **COMPLETE**.

3.18 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 which are received in a particular transaction feed.
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.

3.19 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:

You must not execute the **Cancellation (C1-TXCNC)** batch in the Healthcare domain. This is because the Transaction Cancellation process is not yet supported for the Administrative Services Only (ASO) business.

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 **TFM - 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 **TFM - 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.
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 **Validate Transaction and Derive Price Item (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-TXNPS	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-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
	Thread Count	4 Threads Per CPU
C1-IDENT	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-PDTXN	Maximum Batch Count	5000 Transactions per 16 GB of RAM
	Thread Count	4 Threads Per CPU
C1-DRSUA	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-TXCNC	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-DELBL	Maximum Batch Count	5000 Transactions per 16 GB of RAM