Oracle® Banking Platform Collections

Batch Execution Guide

Release 2.3.1.0.0

E92632-01

December 2017

Oracle Banking Platform Collections Batch Execution Guide, Release 2.3.1.0.0

E92632-01

Contents

3 Technical Recommendations

List of Tables

Preface

This document helps you to understand the sequence in which the batches should be executed. It also helps you to improve the batch performance.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Organization of the Guide
- Related Documents
- Conventions

Audience

This document is intended for the following audience:

- Implementation Team
- Consulting Team
- Development Team

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Organization of the Guide

This document contains:

Chapter 1, "Introduction"

This chapter presents an overview of the Collections module and task flow in Collections.

Chapter 2, "Batch Execution Sequence"

This chapter explains the sequence in which the batches should be executed.

Chapter 3, "Technical Recommendations"

This chapter recommends parameter values for better performance.

Related Documents

For more information, see the following documentation:

- For the configuration details to be done as part of day zero activity, see the Oracle Banking Platform Collections Day Zero Setup Guide

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

Oracle Banking Platform (OBP) Collections lets you integrate with the OBP Core Banking application.

OBP Collections processes all incoming delinquent and non-delinquent (preemptive) accounts from core banking.

This document describes details of the batch processes required as a part of OBP Collections along with batch execution sequence.

Business Process Task flow

Following steps describe the task flow:

1. OBP Collections receives delinquent accounts from different host systems.
2. Cases are created for these delinquent accounts and are tracked through Promise To Pay (PTP).
3. When payments are applied to these accounts, case life cycle tracker brings these cases to closure if outstanding amount for the account is zero.
4. When all cases for an account are closed, OBP Collections marks these accounts as not in Collections.

Features

Following are the features of the batch processes:

- If the 'Batch Business Date' is provided by the user through UI or command line, then batch should take this date and execute, else the default 'System Date' will be used.
- Currently, DateUtility has two methods, 'getPostingDate' and 'getCollectionSystemDateTime'. For 'BATCH' mode the 'getPostingDate' would return the date set as per above point.
- During implementation, you can use 'getPostingDate' to retrieve value passed while batch job submission. This value can be used for 'No Activity' check in No Activity Monitoring Algorithm.
- During implementation, you need to ensure that correct Business or Posting Date is provided while submitting the Job. This can be performed manually through UI or by writing your own script and using it to feed the Auto Scheduler. The onus of this would lie with implementation.

General Instructions for Implementation Team

Following are the general instructions for the implementation team to be followed while implementing:

1. All the batches need to provide the Batch Business Date while submitting the batch, else system date time would be considered for processing.
2. In batch mode, DateUtility.getPosting Date() would return 'batch business Date', if provided on UI, else it would return system date time. This would be applicable to all batches related to collection.
3. In online mode, DateUtility.getPosting Date() would return posting date after fetching it from host.
4. Your script should call DateUtility.getPosting Date() in online mode to fetch OBP posting date and pass the same as every batch parameter.
5. To get the posting date during implementation, you need to follow the below points:
    1. Write a Shell Script -> Call local Java Client -> Call Web service embedding the application service for Date.
    2. Update batch param files.
    3. If there is no application service, then you can put down your own app service which can call Date utility. By default, every app service in OBP is exposed as web service and hence new app service will also have corresponding web service.

2 Batch Execution Sequence

The following table indicates the sequence in which the batches should be executed:

Table 2–1 Batch Execution Sequence

Batches	Batch Sequence
End of Day Batches	Following batches should be run before EOD in the specified order: ▪ PTP Tracking ▪ Cure Monitor ▪ Account abandon monitor ▪ Collection Statistics ▪ Dialer Extract
Beginning of Day Batches	Following batches should be run after EOD in the specified order: ▪ Delinquency Identification for OBP ▪ Move delinquent accounts to RMB feeder tables for OBP ▪ Validate Incoming data from host ▪ Create Entity ▪ Increment DPD ▪ Derived fields ▪ Suspend Activity Monitor ▪ Bulk Contact Creation ▪ Contact Processing ▪ Strategy Monitor ▪ Queue Allocation Monitor ▪ Display Priority Monitor ▪ User Allocation Monitor ▪ Treatment Activity Monitor ▪ Event Manager
Multiple times during the Day Batches	Following batches should be run multiple times during the day in the specified order: ▪ Case Status Automatic Transition Batch ▪ Update Entity ▪ Payment handling ▪ Case Lock

2.1 Delinquency Identification Batch (C1-DELID)

The **Delinquency Identification (C1-DELID)** batch is used to identify delinquent accounts for OBP. The filter conditions passed as parameters to the batch scan all accounts in OBP and mark the account as delinquent when a filter condition is satisfied. The filters must be defined in order of their priority that is, filter with highest priority must be defined first.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Views created on the OBP tables should be available in RMB database schema.

   Delinquency is identified by using data from the following views:

   - FLX_DD_COL_DATA_XF_VW (CASA Accounts)
   - FLX_DD_COL_DATA_TOD_XF_VW (CASA Accounts)
   - FLX_LN_COL_FD_ACCT_VW (TL Accounts)
   - FLX_AC_COL_FD_ACCT_ARS_VW (TL Accounts)

   Updatable views are:

   - FLX_DD_COL_DATA_XF_UPD_ACCT_VW (CASA Accounts)
   - FLX_DD_COL_DATA_XF_UPD_EXTN_VW (CASA Accounts)
   - FLX_LN_COL_ACCT_UPDATE_VW (TL Accounts)

2. Facts based on which the filters are created, must be available in the system.
3. Filters to identify delinquent accounts for OBP data should be created. The product pre-ships five delinquency identification filters as mentioned below:

   - **TL filters**

     - DebitBalance

     - NonPayment

     - PartialPayment

   - **CASA filters**

     - Drawal_LimitPeriod

     - Drawal_ExpiryOfLimit

Dependent Batch

- Move delinquency accounts to Feeder (RMB)
- Incoming Data from Host

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–2 Delinquency Identification Batch (C1-DELID)

Parameter	Mandatory (Yes or No)	Description
Delinquency Filter	Yes	Used to specify the filters that are used to identify delinquent accounts. The filters are defined in order of their priority that is, from highest priority to lowest priority. Multiple filters are separated by a comma. For integration with OBP Host, this filter is product shipped. See the Oracle Banking Platform Collections Day Zero Setup Guide for details of this filter.
Type Of Account	Yes	Used to specify the type of account on which the filters are executed. The valid values are CASA or TL.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.

Post Execution Check / Clean Up

On successful completion of this batch, it updates delinquency flags in host for the accounts filtered by the condition builder. If an account is identified as not delinquent, then this batch process updates the 'Delinquency Flag' as 'N' while the 'Collections Flag' still remains as 'Y'. The 'Delinquency Reason' and 'Delinquency Start Date' are updated as NULL during this stage.

2.2 Move Delinquent Account To Feeder Batch (C1-MVDEL)

The **Move Delinquent Account to Feeder (C1-MVDEL)** batch is used to move the delinquent accounts to Collections feeder tables. This batch should be run once the Delinquency Identification batch completes. It uses a stored procedure (AP_COLL_FDR_DATA_EXTRACT) to move data from OBP to collection feeder tables.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Views created in OBP tables to fetch various details of delinquent customers should be available with associated data. List of views used are mentioned below:
   - FLX_COL_ACCT_DATA_XF
   - FLX_DD_COL_DATA_XF_UPD_ACCT_VW
   - FLX_DD_COL_DATA_XF_UPD_EXTN_VW
   - FLX_LN_COL_ACCT_UPDATE_VW
   - FLX_AC_COL_FD_ACCT_ARS_VW
   - FLX_LN_COL_FD_SCH_VW
   - FLX_COL_ACCT_HRDSHIP_VW
   - FLX_AC_WARNING_IND_COL_XF_VW
   - FLX_PI_COL_FD_ACCT_PER_VW
   - FLX_PI_COL_FD_PER_VW
   - FLX_PI_COL_FD_PARTY_IDENT_VW
   - FLX_PI_COL_FD_PER_NAME_VW
   - FLX_PI_COL_FD_PER_ADDR_VW

- FLX_PI_COL_FD_PER_WARN_IND_VW
- FLX_PI_COL_FD_EMP_PROF_VW
- FLX_PI_COL_FD_CONTACT_PREF_VW
- FLX_LM_COL_FD_COL_ENTITY_VW
- FLX_LM_COL_FD_COLLATERAL_VW
- FLX_LM_COL_FD_COL_PARTY_VW
- FLX_LM_COL_FD_COL_CHRG_VW
- FLX_LM_COL_FD_COL_GRNTR_VW
- FLX_LM_COL_FD_INSR_DTLS_VW

2. Configuration or mapping tables mentioned below should be properly defined:
   - **Source to Collection Class Mapping: CI_HOST_MAIN_CUST** - To identify main customer, financially responsible and guarantor relationship types.
   - **Source to Contract type mapping: CI_HOST_SA_TYPE** - To derive contract type to be used for product class, product group.
   - **Source to product group mapping: CI_HOST_PROD_CL** - To derive customer class, collection class and debt class for product class, product group.
   - **CI_BANK_MST** - List of banks created in host.
   - **CI_BANK_BRANCH_MST** - List of bank branches created in host.
   - **CI_PRIM_NAMETYPE** - Primary name type for host.

Dependent Batch

- Validate Feeder batch
- Create Entity batch

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–3 Move Delinquent Account To Feeder Batch (C1-MVDEL)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.

Post Execution Check / Clean Up

On successful completion of this batch, the data is inserted into the feeder tables.

Verify CI_FDR_PER and CI_FDR_ACCT tables for party and accounts pulled from host.

2.3 Validate Incoming Data from Host Batch (C1-VALFD)

The **Incoming Data from Host (C1-VALFD)** batch is used to validate the records in feeder tables. It uses a stored procedure (AP_COLL_FDR_DATA_VALIDATE) to validate the data.

Validation includes availability of lookup data, admin data and configuration related data. PROCESS_STATUS column in each feeder table is used to identify if the record can be processed. Records that fail validation are marked as 'F' and message category or number will be updated against these records.

Other than validation, this batch updates some columns like division, contract type, and so on (derived from configuration tables) in CI_FDR_ACCT table.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Delinquent accounts should be available in the feeder tables
2. Admin data and lookups

Dependent Batch

- Create entity batch

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–4 Validate incoming Data from Host Batch (C1-VALFD)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.

Post Execution Check / Clean Up

On successful completion of this batch, feeder data is marked failed for the records that fail validation.

Check PROCESS_STATUS, MESSAGE_CAT_NBR and MESSAGE_NBR column in CI_FDR_PER table.

2.4 Create Entity Batch (C1-CRENT)

The **Create Entity (C1-CRENT)** batch is used to select records from feeder tables and create entries for delinquent accounts.

Entity creation involves following entity objects:

- Party / Person (and related data)
- Account (and related data)
- Party account relationship

- Mainline
- Collateral
- Insurance

Setup Prerequisites

Following is the prerequisite for the setup:

1. Validated data should be available in feeder tables. Records with PROCESS_ STATUS = 'P' will be processed.

Dependent Batch

- Strategy Monitor batch
- Increment DPD batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–5 Create Entity Batch (C1-CRENT)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, party and account data is created in Collections. TO DO is created for failed records from feeder tables. Records for successfully created data are deleted from feeder tables.

Verify CI_PER and CI_ACCT for party and accounts created in Collections.

2.5 Update Host Data Batch (C1-UPENT)

The **Update Host Data (C1-UPENT)** batch is used to update the Collections tables based on the updates received from host systems in the feeder tables.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Data is available in feeder tables with updates from hosts. CI_FDR_HOST_ UPDATES table should have list of table names whose updates have come from host.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–6 Update Host Data Batch (C1-UPENT)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful execution of this batch, updated records are moved to the collection tables and feeder table records are deleted.

Records in CI_FDR_HOST_UPDATES should be deleted for successfully processed updates.

2.6 Increment Day Past Due Batch (C1-INCDP)

The **Increment Day Past Due (C1-INCDP)** batch is used to increment the DPD for an account if no updates are received from the host system. In addition, this batch also updates the bucket value for all accounts even if the accounts DPD is not incremented.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Bucket configuration should be available in CI_COLLECT_BUCKET_MST table.

Dependent Batch

- Display Priority Monitor

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–7 Increment Day Past Due Batch (C1-INCDP)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Exclude Source Host ID	No	Used to specify the source host IDs that are to be excluded from the batch processing. Multiple source host IDs are placed in single quotes and are separated by comma. For example, 'A', 'B', 'C'. If you leave this field blank, then the batch will be executed for all the host IDs.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful execution of this batch, DPD values are updated or are incremented by the difference between system date and last DPD update date. This can be verified in CI_PARTY_COLLECT table.

Also, the account buckets are updated and can be verified in CI_ACCT_BUCKET table.

2.7 Feeder Payment Batch (C1-FDPAY)

The **Feeder Payment (C1-FDPAY)** batch is used to create payments that have failed during online payment creation and are received from the host system.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Account for which payment is received from host must be available in Collections.
2. Collections case should be in open status and at least one active contract should be present on account.
3. FT freeze algorithm should be attached to adjustment type.

Dependent Batch

- PTP Tracking

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–8 Feeder Payment Batch (C1-FDPAY)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, payment is marked, accounting entries are passed, and overdue balances are updated. Feeder records are deleted.

Payment records successfully created can be verified from CI_PAY, CI_PAY_EXTN, CI_ PAY_EVENT tables.

2.8 Derived Field Batch (C1-DRFLD)

The **Derived Field (C1-DRFLD)** batch is used to update columns derived from Collections data. The following derived fields are updated:

- Secured Y/N
- PPI Insured Y/N
- LMI Insured Y/N
- Co-borrowers Exist Y/N

Any other business logic to be derived other than the ones mentioned above can be done using an algorithm created on DerivedFieldAlgorithmSpot and passing it to the batch.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Incoming data from host must be available in Collections.
2. Derived field exclusion setup (CI_DERIVED_EXCLUSION) should be available.

Dependent Batch: Not Applicable

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–9 Derived Field Batch (C1-DRFLD)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Algorithm Code	No	Used to specify an algorithm name, which will be invoked by the batch process for additional logic to be performed.

Post Execution Check / Clean Up

On successful completion of this batch, the derived fields are updated. Verification can be done in CI_ACCT_EXTN table.

2.9 Promise To Pay Monitor Batch (C1-PTPM)

The **Promise to Pay Monitor (NGPPPM)** batch is used to monitor pending and active PTPs. This batch monitors the PTPs payer and the service-level agreement of the same debt class. This batch process also determines if a PTP has been kept, broken, or active.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Person, Account, and Contract must be created for the delinquent accounts.
2. PTP should have been created on accounts for processing.

Dependent Batch

- For EOD, Strategy monitor is dependent.

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–10 Promise To Pay Monitor Batch (NGPPPM)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, PTP status is updated for accounts. Verification can be done in CI_PTP table.

2.10 Cure Monitor Batch (C1-FINCO)

The **Cure Monitor (C1-FINCO)** batch is used to close cases and move account out of Collections.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Condition builder must be created to filter the account to be finalized.
2. Host API should be available to mark account Not in Collections.

Dependent Batch

- Strategy Monitor Batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–11 Cure Monitor Batch (C1-FINCO)

Parameter	Mandatory (Yes or No)	Description
Filter Name	Yes	Used to specify the filter ID to fetch the accounts that need to be cured.
Promise To Pay Cancellation Reason	Yes	Used to specify the cancellation reason to cancel PTP.
Cure Monitor View	Yes	Used to view the name of cure monitor.
Cure Entity Algorithm Code	No	Used to specify the algorithm to close the contract if there are no open cases on an account.

Table 2–11 (Cont.) Cure Monitor Batch (C1-FINCO)

Parameter	Mandatory (Yes or No)	Description
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, accounts are marked as Not in Collections in host and in Collections. Contract will be stopped, case will be moved to final status and active PTP is cancelled.

Verification can be done from CI_CASE, CI_PTP and CI_ACCT_EXTN tables.

2.11 Account Abandon Monitor Batch (WRITEOFF)

The **Account Abandon Monitor (WRITEOFF)** batch is used to request for a complete account write-off either in the Straight Through Processing (STP) or in Non-STP (Manual) modes.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Set up Rule Engine and Condition builder. Derived facts will be used to retrieve write off type from rule engine.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–12 Account Abandon Monitor Batch (WRITEOFF)

Parameter	Mandatory (Yes or No)	Description
Account Write Off View	Yes	Used to filter accounts for write off.
Filter Name	Yes	Used to specify the filter name configured in the Filter Definition screen (Back Office > Rules) to filter accounts that needs to be written off.
Rule Name	Yes	Used to specify the rule name configured in the Rule Author screen (Back Office > Rules) to provide write off type.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, write off request date is updated in Collections and request is sent to host for account write off. Verification can be done in the CI_ ACCT_EXTN table.

2.12 Strategy Monitor Batch (C1-CSMB)

The **Strategy Monitor (C1-CSMB)** batch is used to create cases for delinquent accounts based on the case type retrieved from the rule. If TAM details are configured in the rule, these will be updated for the new case being created. This batch creates only account level cases.

The possible output expected from the rule is:

- Collection Strategy (Case Type)
- Re-evaluation days
- TAM Matrix ID
- TAM_REVIEW_DT

Setup Prerequisites

Following are the prerequisites for the setup:

1. Rules / Rule set for case creation must be defined in the rules engine.
2. Case types with status should be defined. Algorithms should be attached to case status.
3. PTP Tracking batch should be complete.
4. CI_ADM_RVW_SCH should have records with NEXT_CR_RVW_DT <= system date

Dependent Batch

- Case Status Automatic Transition batch
- Queue Allocation batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–13 Strategy Monitor Batch (C1-CSMB)

Parameter	Mandatory (Yes or No)	Description
Rule to retrieve case type and re-evaluation days	Yes	Used to specify the Rule Set ID or Rule ID to retrieve case type and re-evaluation days.
isRuleSet flag	Yes	Used to specify a value to call the rule set. Enter 'Y' to call a Rule Set or enter 'N' to call a Rule.
Rule Fact Population Algorithm	Yes	Used to specify an algorithm code for rule fact population. (C1-BRLSR algorithm type pre-shipped)

Table 2–13 (Cont.) Strategy Monitor Batch (C1-CSMB)

Parameter	Mandatory (Yes or No)	Description
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, cases are created for delinquent accounts. If CSAM algorithm is attached to the case status, then it will trigger and take actions as defined in the CSAM admin configuration.

Cases created can be verified in CI_CASE, CI_CASE_EXTN, CI_CASE_PARTY and CI_CASE_ASSOCIATION tables.

2.13 Case Status Automatic Transition Batch (CASETRAN)

The **Case Status Automatic Transition (CASETRAN)** batch is used to call the algorithm that determines whether a case state should be transitioned to a new state.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Case type status should be defined.

Dependent Batch

- Queue Allocation Monitor batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–14 Case Status Automatic Transition Batch (CASETRAN)

Parameter	Mandatory (Yes or No)	Description
Restrict To Case Type Code	No	Used to specify the case type to limit the process to cases of this type.
Restrict To Cases in Status Code	No	Used to specify the case status code to limit the process to cases in this status.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before the run terminates.
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

Cases will be moved to the next status. Verify this from CI_CASE table.

2.14 Queue Allocation Monitor Batch (C1-ALOCM)

The **Queue Allocation Monitor (C1-ALOCM)** batch is used to allocate cases to queues. Cases which are to be allocated to the queues are filtered by the condition builder attached to the allocation group and are assigned in round-robin method to the queues that are linked to the allocation group.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Queues and Allocation Groups should be configured.
2. Overflow queue should be attached to allocation groups.
3. Queue allocation algorithm should be attached to allocation group.
4. Condition builder to filter accounts for allocation should be attached to allocation group.

Dependent Batch

- User Allocation batch
- Display Priority Monitor batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–15 Queue Allocation Monitor Batch (C1-ALOCM)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, cases are allocated to queues.

Verification can be done in CI_QUEUE_ASSIGN table.

2.15 Collection Statistics Batch (C1-COLST, C1-CLDCS)

The **Collection Statistics (C1-COLST)** batch is used to create statistics from collection data. The following columns are updated:

- PTP's Broken (Derived from CI_PTP table)
- PTP's Kept (Derived from CI_PTP table)
- Consecutive PTP's Broken (Derived from CI_PTP table)
- Peak OD days (Derived from CI_PARTY_COLLECT.DAYS_PAST_DUE)
- PTP's taken (Derived from CI_PTP table)
- Last acted agent (Derived from CI_CASE_LOG.USER_ID)

C1-CLDCS batch is used to generate delinquency and cycle string.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Account and Customer data should be available in Collections.
2. Bucket Admin configuration should be available.

Dependent Batch: Not Applicable

Multi-threaded:

C1-COLST: No

C1-CLDCS: Yes

You can specify the following parameters while executing this batch:

Table 2–16 Collection Statistics Batch (C1-COLST, C1-CLDCS)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before the run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

Statistics data will be updated based on data available. Verification can be done in CI_COLLECT_STAT table.

2.16 Suspend Activity Monitor Batch (C1-SUSMN)

The **Suspend Activity Monitor (C1-SUSMN)** batch is used to identify activities that must be stopped or suspended when a set of conditions is satisfied for a customer or an account. This batch also removes the Suspend flag if conditions are not satisfied.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Condition builder must be set up and linked to suspend activities in suspend activity admin setup.

Dependent Batch

- Strategy Monitor batch
- Queue Allocation Monitor batch
- Case Status Automatic Transition batch
- Bulk Contact Creation batch

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–17 Suspend Activity Monitor Batch (C1-SUSMN)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before the run is terminated.
Suspend Activity View Name	Yes	Indicates the name of the view, based on which the records are selected for suspension. CI_SUSPEND_ACTIVITY_VW view is pre-shipped from product.

Post Execution Check / Clean Up

On successful completion of this batch, flags are set for accounts or customers that satisfy the filter conditions defined in the suspend activities.

Verification can be done from CI_SUSP_ACCT_ACTY and CI_SUSP_PER_ACTY tables.

2.17 Bulk Contact Creation Batch (C1-BLKCC)

The **Bulk Contact Creation (C1-BLKCC)** batch is used to generate contacts (of type Letters, E-mail, and SMS) in volume based on certain conditions.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Bulk Contact admin configuration should be defined and condition builder to filter accounts should be attached to each record.

Dependent Batch

- Contact Processing batch

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–18 Bulk Contact Creation Batch (C1-BLKCC)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before the run terminates.
Bulk Contact View Name	Yes	Used to specify the view name to be queried for list of accounts. CI_BULK_CONTACT_VW view is pre-shipped from product.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, customer contacts are created for contact processing batch to generate associated letters.

Verification can be done from CI_CC table.

2.18 Contact Processing Batch (C1-CNTPR)

The **Contact Processing (C1-CNTPR)** batch is used to select un-processed customer contacts and invoke the letter template extract algorithm, which generates associated Letters, SMS, or e-mails.

Setup Prerequisites

Following is the prerequisite for the setup:

1. For each contact type, an extraction algorithm should be associated from contact type configuration screen.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–19 Contact Processing Batch (C1-CNTPR)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Process Failed Contacts	No	Used to specify the option to process failed contacts. Enter 'Y' to process failed contacts and 'N' to cancel processing failed contacts.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, all contacts that have been processed are marked as processed.

Verification can be done in CI_CC table. Records failed during processing can be verified from NON_DELIVERY_REASON in CI_CC table.

2.19 Display Priority Monitor Batch (C1-PRMON)

The **Display Priority Monitor (C1-PRMON)** batch is used to set priority of a case. This is used while displaying cases on the user interface.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Define **Display Order** and **Arranged By** for queue from queue details admin setup.
2. Define work list required at queue level.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–20 Display Priority Monitor Batch (C1-PRMON)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, the display order of all open cases is set. Verification can be done in the CI_QUEUE_ASSIGN table.

2.20 User Allocation Monitor Batch (C1-USALC)

The **User Allocation Monitor (C1-USALC)** batch is used to allocate cases to users, teams, or vendors associated within queues.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Allocation group and queues should be configured.
2. User allocation algorithm should be mapped to queues.
3. Collection users, teams or vendors should be mapped to queues in queue details screen.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–21 User Allocation Monitor Batch (C1-USALC)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, cases are allocated to users, teams, or vendors.

Verification can be done in CI_USER_ASSIGN and CI_VENDOR_ASSIGN for users and vendors respectively.

2.21 Treatment Activity Monitor Batch (C1-TRMON)

The **Treatment Activity Monitor (C1-TRMON)** batch enables the Collections system to provide a framework that helps in the following activities:

- Associate cases to Treatment Activity Monitor decision matrix.
- Invoke Treatment Activity Monitor decision matrix.
- Perform actions based on the Treatment Activity Monitor decision matrix outcomes.
- Modify or Remove Treatment Activity Monitor decision matrix associations.
- Batch picks up cases which are not on hold and where TAM_REVIEW_DT <= business date.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Cases should be available with TAM MATRIX ID's.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–22 Treatment Activity Monitor Batch (C1-TRMON)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates.
Case Type	No	Used to specify the case type to be used for treatment activity monitor batch.
Treatment Activity Monitor Algorithm Code	Yes	Used to specify the TAM algorithm code to be used for processing of all the selected cases.
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

Based on the condition defined in TAM algorithm, necessary actions will be taken. TO DO will be created if there is an error while processing the algorithm.

2.22 Event Manager Batch (C1-EVENT)

The **Event Manager (C1-EVENT)** batch is used to execute all events attached to condition category and perform all outcomes attached to events.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Validate events are configured for condition category.

Dependent Batch: Not Applicable

Multi-threaded: No

You can specify the following parameters while executing this batch:

Table 2–23 Event Manager Batch (C1-EVENT)

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

Post Execution Check / Clean Up

On successful completion of this batch, all outcomes attached to condition events are executed, log entry is created for successful executed outcomes in the Event Log table with status as SUCCESS. Log entry is also created for failed records in Event Log table with status as FAILED.

2.23 Case Lock Batch (C1-CSCL)

While a user is working on a case, it gets locked for that user. In some scenarios such as a server crash or a browser closure, the case gets locked for a user at liberty.

In such scenarios **Case Lock** batch creates a task for Supervisor who has locked a case for more than stipulated time.

Setup Prerequisites

Following are the prerequisites for the setup:

1. Feature Configuration (C1-TDTYPLOCK) should be created before running this batch.
2. CI_QUEUE_TODO_ROLE table should have mapping of Role and Queue and subsequent mapping of role and To Do Type should be present in To Do Type Role mapping table.

Dependent Batch: Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–24 Case Lock Batch (C1-CSCL)

Parameter	Mandatory (Yes or No)	Description
Thread Pool Name	No	Used to specify the thread pool on which you want to execute the batch.
Time (in minutes)	Yes	Used to compare the time with the locked time
User	No	Optional to filter cases created by entered user
Case Type	No	Optional to filter cases created by entered Case Type
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up

On successful completion of this batch, TO DO for supervisor of a user who has locked a case for more that stipulated amount of time will be created in Collections.

2.24 Dialer Extract Batch (C1-DIEXT)

This batch program will fetch account ID's from views satisfying the condition given in the filter. This filter needs to be defined in OBP. After fetching these records, batch program will insert it into staging area for Dialer.

Setup Prerequisites

Following is the prerequisite for the setup:

1. Lookup (C1_EXTRACT_STATUS, CHANNEL_TYPE) should be created before running this batch.

Dependent Batch:

- Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–25 Dialer Extract Batch (C1-DIEXT)

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
MAX-ERRORS	No	Enter a value here to override the maximum number of errors allowed before run terminates
VIEW-NAME	Yes	View to Fetch acct_id based on extract conditions
OVERRIDE-FLAG	No	Optional. Expected values Y/N. Default value is N. The flag is used to control insert/updates in staging table
EOD-FLAG	Yes	Values (Y/N). If this flag is 'Y' then records in the staging area should be deleted, else if flag is 'N' then records should not be deleted
EXTRACT-TYPE	Yes	If provided then all set of conditions (filters) specified in EXTRACT-CONDITIONS will be executed in order of priority defined in the batch parameter
EXTRACT-CONDITIONS	Yes	Used to provide filter condition
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up:

On successful completion of this batch, records will be inserted in the CI_DIALER_EXTRACTS table. On failure TO DO will be created for failed accounts.

2.25 Dialer Results Upload (C1-DLRRS)

Dialer and IVR systems provide a response file for all the calls made during the course of the day. This information needs to be uploaded in RMB, that is the Staging area. This batch program will read the data from the staging area and upload the results.

Setup Prerequisites

Following is the prerequisite for the setup:

1. A new master (admin) screen is needed for mapping the IVR and dialer results. There will be one set of configuration for Dialer and one for IVR file. This will be done from the same screen.

Dependent Batch:

- Not Applicable

Multi-threaded: Yes

You can specify the following parameters while executing this batch:

Table 2–26 Dialer Results Upload (C1-DLRRS)

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
Override Maximum Errors	No	Enter a value here to override the maximum number of errors allowed before run terminates
Execution Status	No	Enter a value ('E' or 'ERROR') to execute all the records in staging area having status as 'ERROR'
Thread Count	No	Recommended thread count is 4 per CPU.

Post Execution Check / Clean Up:

- For Entity-Contact, contacts will be created and action results will be logged against those contacts.
- For Entity-Case, action results will be logged against all the cases related to a customer or account.
- On failure TO DO will be created for failed records.

3 Technical Recommendations

This section recommends parameter values to achieve maximum performance. This will vary with different hardware set. The actual performance would depend on the number of CPUs and RAM available on the application server, and many other hardware parameters.

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

To improve the overall batch performance of 64-bit application server on Linux 64-bit environment, we recommend you to make changes in the following files:

Table 3–1 Recommendations for Files to be Modified

File Name	Change From	Change To
hibernate.properties	hibernate.c3p0.timeout = 300	hibernate.c3p0.timeout = 600
threadpoolworker.sh	MEM_ARGS="-Xms512m -Xmx1024m -XX:MaxPermSize=768m "	MEM_ARGS="-Xms512m -Xmx4096m -XX:MaxPermSize=768m "

Note: These settings are not applicable for 32-bit application servers on Windows, AIX, and Linux platforms. Oracle Revenue Management and Billing is already tuned to achieve maximum performance on 32-bit application servers.

Recommended Thread Count for a Batch:

Thread count for each batch that supports multi threading should be as per the number of available CPU and volume of data being handled. Recommended value is 4 Thread per CPU should be used.

Common Input for All Batches:

E-mail Address: This will be the E-mail ID of the person to whom notification is to be sent in case of TO DO created by batches.

User ID: This will be a valid user if in collections and will be used as user id executing the batch.

This user ID should also be mapped to OFFLINE_CHANNEL_USER_NAME property in /config/properties/ CredentialPropertyStore.properties file.