Oracle Financial Services Revenue Management and Billing Cloud Service, Premium Edition

OR

Oracle Insurance Revenue Management and Billing Cloud Service, Premium Edition

Version 5.1.0.0.0

Live Operations Guide

Revision 1.1

F83991-01

July 2023

Oracle Financial Services Revenue Management and Billing Cloud Service, Premium Edition/Oracle Insurance Revenue Management and Billing Cloud Service, Premium Edition Live Operations Guide

Note: The above two products are collectively referred as Oracle Revenue Management and Billing Cloud Service, Premium Edition throughout this document.

F83991-01

Preface

About This Document

This document provides guidelines regarding live operations of Oracle Revenue Management and Billing Cloud Services.

Intended Audience

This document is intended for the following audience:

- End-Users
- Implementation Team

Organization of the Document

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

Section No.	Section Name	Description
Section 1	Cloud Services Live Operate Services	Provides information related to the live operate services for Oracle Revenue Management and Billing Cloud Services.
Section 2	Cloud Services Live Operational Guidelines	Describes the operational guidelines for the Oracle Revenue Management and Billing Cloud Services.
Section 3	Data Fix with Plug-in Driven Batch	Describes the guidelines that apply when the plug-in driven batch processes are used to fix the data issues.
Section 4	Troubleshooting	Provides troubleshooting guidelines for Oracle Revenue Management and Billing Cloud Services.

Conventions

The following conventions are used across the document:

Convention	Meaning
boldface	Boldface indicates graphical user interface elements associated with an action, or terms defined in the text.
italic	Italic indicates a document or book title.
`monospace`	Monospace indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or information that an end-user needs to enter in the application.

Acronyms

The following acronyms are used in this document:

Acronym	Meaning
OUAF	Oracle Utilities Application Framework
CMA	Content Migration Assistant
CSF	Cloud Service Foundation
CSM	Customer Success Manager
ORDS	Oracle REST Data Services
SaaS	Software-as-a-Service
SR	Service Request
TPW	Threadpool Workers

Related Documents

You can refer to the following documents for more information:

Document Name	Description
Oracle Revenue Management and Billing Cloud Service, Premium Edition Administration Guide	Explains how to manage the user accounts and their access for Oracle Revenue Management and Billing Cloud Services (ORMBCS) using Identity and Access Management with or without identity domains on Oracle Cloud Infrastructure (OCI).
Oracle Revenue Management and Billing Cloud Service, Premium Edition Frequently Asked Questions Guide	Lists various frequently asked questions (FAQs) regarding the implementation and operations of Oracle Revenue Management and Billing Cloud Services.
Oracle Revenue Management and Billing Cloud Service, Premium Edition Implementation Guide	Provides information on how to implement the Oracle Revenue Management and Billing Cloud Service.
Oracle Revenue Management and Billing Cloud Service, Premium Edition Operations Guide	Provides information regarding different types of service requests (SRs) customers can submit to the Oracle Revenue Management and Billing Cloud Operations team during implementation and operations of the Oracle Revenue Management and Billing Cloud Services.

Access to Oracle Support

Oracle customers that have purchased support 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.

Change Log

Revision	Last Update	Updated Section	Comments
1.1	01-Aug-2023	Cover Page	Updated Information
		Copyright Notice	Updated Information
		Preface	Added Information

Contents

1. Cloud Services Live Operate Services

This section provides information regarding live operate services for Oracle Revenue Management and Billing Cloud Services. It contains the following topic:

- Live Operate Services

1.1 Live Operate Services

Live Operate services are services that Oracle provides to assist you with the ongoing operation of supported Oracle Revenue Management and Billing Cloud Services, once live and in production. If Live Operate Services are included in your Oracle Revenue Management and Billing Cloud Service, it will be stated in the relevant services descriptions.

The scope of the Live Operate Services provided by Oracle is limited and currently includes the following:

Oracle will work to:

- Assign a Customer Success Manager ("CSM") to manage delivery of the Oracle Revenue Management and Billing Cloud Service. The CSM will be the primary point of contact, will act as the first point for escalations, will monitor the progress of any related service requests ("SR") and confirm that all Oracle Revenue Management and Billing Cloud Services are performed in accordance with targets.
- Provide access to information about incidents and changes made to any of your Oracle Revenue Management and Billing Cloud Service environments.
- Upon request, create and deliver a Quarterly Status Report summarizing the activities undertaken over the past quarter and those scheduled as major future events.
- Upon request, offer quarterly meetings with customer/implementation oversight team to present, review and discuss the Quarterly Status Report.
- Monitor infrastructure and application availability and resolve any incident that is within Oracle's scope or responsibility.
- Provide primary Help Desk Services from 24*5 your local time on Oracle working days.
- Respond to severity 1 incidents 24 hours per day, 365 days per year.
- Provide incident management and problem management services for events related to Oracle Revenue Management and Billing Cloud Services, including:
    - Analyzing issues and resolving any incidents that are within Oracle's scope or responsibility.
    - Escalating any issues not within Oracle's scope or responsibility to you for resolution.
    - Provide infrastructure logs to assist in the diagnosis and remediation of incidents within your scope or responsibility. NB: these logs may be scrubbed for security purposes and other sensitive data.
    - Executing scripts authored by you to resolve data issues, but only to the extent such execution is necessary and the circumstances necessitating execution are exceptional.

        - Scripts submitted for execution to correct data issues will be subject to review for compliance with Oracle's Software Security Assurance standards. Scripts that do not comply will be returned for correction; resubmission will require a further compliance review with approval before execution will be scheduled.
        - There is a limit of two (2) script reviews in any calendar month.

Customers remain completely responsible for all other aspects of using or otherwise operating the cloud service. Examples of customer obligations in this regard would be to:

- Assign a manager to act as the primary conduit for all issues relating to the delivery of these services.
- Provide Oracle with the dates for key business and technology events (i.e., testing calendar, refresh schedule etc.) at least thirty (30) days prior to the event.
- Assign appropriate resources to support your activities.
- Support scheduling of meetings as required.
- Participate in all scheduled meetings.
- Manage all activities, including:
    - Prioritizing work activities.
    - Providing coordination across any teams and/or interested parties external to Oracle.
    - Escalation of issues to the Oracle CSM in a timely manner.
    - Providing any information required to progress service requests. Oracle is not responsible for meeting any agreed (or stated) targets or objectives if required or requested information is not forthcoming in a timely manner.
    - Assigning appropriate resources to conduct all required acceptance testing for all software packages delivered.
    - Accepting all software packages prior to promotion to the Production environment.
    - Maintaining a contact list for all persons performing governance functions.
    - Participating in Oracle's internal quality assurance processes on a mutually agreed schedule.
- Manage the Oracle Revenue Management and Billing Cloud Services, including:
    - Defining and (as necessary) modifying batch schedules to meet business needs.
    - Monitoring batch and interface processes to confirm and verify completion.
    - Defining, implementing, testing, and deploying any changes to the configuration and extensions required to:
        - Resolve incidents or problems within the customers scope or responsibility
        - Meet evolving business needs
        - Conform with upgrades, patches or any other Oracle Revenue Management and Billing Cloud Services requirements or prerequisites
- Notify Oracle of any incidents detected.
- Action each SR in accordance with the relevant Oracle Cloud Hosting and Delivery Policies.
- Establish the method of communication between the incident management team and the Oracle team for each incident.

- Redirect any SRs to the correct (non- Oracle) support team if the SR is not within the scope of the Oracle Revenue Management and Billing Cloud Services.
- Assist the Oracle team with SR analysis of data originating outside the applications.
- Correct business data as reasonably requested by Oracle to achieve closure of an identified problem.
- Monitor customer network and network connections to ensure sufficient bandwidth and performance.
- Conduct all functional, regression, performance, system integration, acceptance and/or user acceptance testing as appropriate, and accept and approve all software changes, prior to the production roll out of any change, in accordance with any agreed-on release management process, for any change resulting from:
  - An incident remedy or fix, or
  - Any service pack update or upgrade
- Provide authorizations and/or validations for data correction and/or adjustments to the supported environments.
- Apply configuration changes that are within the customer's or implementation’s scope as recommended by Oracle to prevent recurring incidents.
- At Oracle's request, provide approval through the environment change management process for the installation of fixes to prevent recurring incidents.
- Initiate documented escalation processes when the urgency of the incident has increased due to business requirements.
- Provide incident support where the remediation is within Oracle's scope or responsibility, specifically:
  - Provide Oracle with access to the user who reported the problem.
  - Provide all information as reasonably requested by Oracle and participate in diagnostics.
  - Provide a test case and access to an environment where the SR incident is reproducible.
- Provide Oracle, as necessary, with access to any other support teams.
- Acknowledge that the supplied break-fix resolves the incident or problem and accept the SR closure.
- Provide internal approvals to Oracle before the migration of any change recommended by Oracle into a production environment.
- When an incident or problem is determined by Oracle to be unrelated to an Oracle Revenue Management and Billing Cloud Services or to be caused by data problems arising from data conversion or some other error, a separate, paid contract may be required to cover Oracle's time spent investigating and, if relevant, correcting the problem at Oracle's prevailing time and materials rates, and to reimburse Oracle for all reasonable out of pocket expenses.

2. Cloud Services Live Operational Guidelines

This section provides guidelines around the operation of Oracle Revenue Management and Billing Cloud Services once configured to a customer's requirements. It contains the following topics:

- Batch Job Submission
- Batch Job Scheduling
- Level of Service Batch Monitoring
- Code and Configuration Migration
- Customer Facing Alerts
- Data Cloning/Subsetting
- Limits on Time or Size of Certain Services
- Server Logs - Online and Batch

2.1 Batch Job Submission

Much of the heavy processing in Oracle Revenue Management and Billing Cloud Services is done via batch processing, so it's critical to set up batch jobs to run in the most efficient way, with correct parameters.

Key rule: Run all batches with a **Commit Frequency** of 1. This setting is optimal given the way the application's hibernate entity cache works.

Batch jobs can be submitted in one of the following ways:

- Using a manual or timed submission of a batch job using the application base functionality
- Using Oracle DBMS Scheduler to schedule and submit jobs
- Using an on-premise customer batch job scheduler

Refer to the **Background Processes** section in the *Oracle Utilities Application Framework Administrative Guide* or *Oracle Revenue Management and Billing Online Help* for more information.

2.2 Batch Job Scheduling

This section describes how to use Oracle DBMS Scheduler for scheduling batch jobs.

2.2.1 Using Oracle DBMS Scheduler

The Oracle DBMS Scheduler is provided with Oracle Revenue Management and Billing Cloud Services and is the default job scheduling option.

Refer to the **Batch Scheduler Integration** section in the *Oracle Utilities Application Framework Administrative Guide* or *Oracle Revenue Management and Billing Online Help* for more information.

However, note that access to the DBMS Scheduler via SQL using SQL Developer is not allowed in the cloud.

The Oracle Utilities Application Framework provides various REST services to directly interact with the DBMS scheduler. While batch operations use the underlying services of these APIs to interact with the DBMS scheduler, there are certain restrictions in calling the REST services from outside the system or via a third party integration. Review the *Batch Scheduler Integration for Oracle Utilities Application Framework (Document ID: 2196486.1)* article on My Oracle Support for more information.

A copy of the batch job stream definitions created in batch operations is kept in the Oracle Utilities Application Framework along with publishing them to DBMS scheduler. Any changes to such batch job stream definitions such as step changes, schedule changes, and so on must be done through the batch operations user interface. The Oracle Utilities Application Framework REST services should not be used directly for changing the definitions as this can cause the Oracle Utilities Application Framework job stream definitions go out of sync with the definitions inside the DBMS scheduler. This can result in a risk of losing historical data tracking, logging, risk of losing definitions and re-work etc. It is advised that only those Oracle Utilities Application Framework DBMS Scheduler REST services that pertain to handling job stream runs, not definitions be used from outside of the system or via third party integrations. Such services involve querying on the job stream run status, details, make adhoc submissions of the job stream run, canceling a job stream run and getting past job stream runs.

Cloud customers can use a set of REST APIs to set up and manage their batch scheduling using the Oracle DBMS Scheduler. The available DBMS scheduler APIs can be viewed by searching for Business Services that start with F1-DBMS.

2.3 Level of Service Batch Monitoring

The Oracle Utilities Application Framework provides a Health Check feature that reports on a configurable set of 'Level of Service' algorithms, which can check on various conditions of particular Batch Controls and Batch Job Streams. Results of the Health Check are given in this form (like http return codes):

- 200 - All Checks Successful (i.e., no warnings, no errors)
- 203 – Warning - Non-Critical Function Degraded
- 500 – Error - One or More Critical Functions Degraded

The Health Check can be brought up online at any time and can also be polled regularly from outside of the service via the Inbound Web Service F1-HealthCheckREST.

Level of Service Algorithms are available at the Batch Control as well as Batch Job Stream level, most of which can be set up with parameters to fine tune the error and warning conditions. The delivered algorithm types are:

Level of Service Algorithms – Batch Control:

- F1-BAT-ERLOS: Report Batch Jobs in Error
- F1-BAT-LVSVC: Evaluate Error Count and Time Since Completion
- F1-BAT-RTLOS : Compare Total Batch Run Time to Threshold
- F1-BAT-TPLOS: Compare Throughput to Threshold
- K1BATJNSXM: Batch Job not Started in X Minutes
- K1BATJPLNR: Job Processed Low Number Of Records
- K1BATLOSRTL: Batch Job ran too long (Relative Run)
- K1BATLOSTTL: Batch Job throughput too low (relative run)

Level of Service Algorithms – Batch Job Stream:

- K1BJSDJNSXM: Batch Job Stream Not Started in X Minutes
- K1BJSDJRTL: Job Stream Ran Too Long (Relative Run)
- K1-BJSD-LVSV: Batch Job Stream Has Failed

Refer to the **Service Health Check** section in the *Oracle Utilities Application Framework Administrative Guide* or *Oracle Revenue Management and Billing Online Help* for more information.

2.4 Code and Configuration Migration

All automated configuration migration between environments should be done using Content Migration Assistant (CMA) provided with the Oracle Utilities Application Framework. An automation option for configuration migration, called the Process Automation Tool, is supplied through Cloud Service Foundation (CSF) included in all Oracle Revenue Management and Billing Cloud Services.

Note: Content Migration Assistant (CMA) can only be used to migrate data between environments on the same cloud service release version. For example, data exported from a 21C environment can ONLY be imported into a different 21C environment.

2.4.1 What is Cloud Service Foundation?

Cloud Service Foundation (CSF) is an add-on companion product for Oracle Revenue Management and Billing Cloud Services that provide automation for various generic processes, such as configuration migration. It is installed on top of the Oracle Utilities Application Framework in the same primary application installation in the cloud.

Infrastructure Process Types are provided out-of-the-box to support CMA Accelerator Load and CMA Migration processes (extract from one cloud environment and import into target). The Process Automation Tool provides several migration requests with the base package that orchestrate migration of objects based on appropriate criteria. Alternatively, you can create custom migration requests to select appropriate records based on specific business requirements. To see the migration requests provided by the product, log in to the relevant cloud service application and navigate to the migration request page by selecting **Admin**, then **Implementation Tools**, then **Migration Request**, then **Search**.

2.5 Customer Facing Alerts

Customers must be notified if certain errors or issues arise that will require attention. The most common example is batch alerts.

Customers need to know if important batch processes have failed or have not performed that work they were supposed to do. This could be critical for regular nightly batch processes but is also useful for daily or other scheduled batch processing. Instead of manual monitoring of important processes to make sure they worked as planned, the system can respond to a REST call that inquires about the overall status of the system processing. That service is called System Health Check and support both batch related checking (via the Level of Service algorithms that are plugged into Batch Controls) as well as batch job stream checking (via the Level of Service algorithms that are plugged into batch job stream definition).

In addition to the system health check service, an external probe can be set up that will invoke the REST call to the system from time to time and initiate an email notification to a configurable set of email addresses so that customers don't have to do it themselves manually.

2.6 Data Cloning/Subsetting

Data subsetting is the process of moving data from the production environment to other environments (such as TEST or DEV environment). There are two processes for data subsetting - full-volume and partial-volume.

2.7 Full Volume Clone

The TEST environment is a full-sized environment and can host a complete copy of the production database. A service request should be submitted to request the refresh of TEST from PROD. The two environments must be at the same version/patch level.

After the data refresh, any configurations and scripts that were in TEST and not in production will need to be re-applied to TEST.

The full volume clone can also be used to migrate a 'gold' configuration environment (DEV-sized) to a Test environment or to Production during implementation.

See the *Oracle Revenue Management and Billing Cloud Service, Premium Edition Operations Guide* for more information about submitting data cloning service requests.

2.8 Partial Volume – Subset

The DEV environment is not a full-sized environment and can only hold a subset of the production database. Content Migration Assistant may be used to perform a targeted migration of selected master entities and their related transactional data from one environment to another. For example, migrating a subset of accounts and their related data for testing purposes.

The overall volume of all business entities to migrate in a single data set should be reasonably sized. Migrating too much data may reach physical and performance limitations of the tool.

2.9 Limits on Time or Size of Certain Services

Oracle Revenue Management and Billing Cloud Services have some time and data size limits. In the online application, pages have a default timeout limit of two minutes. This is to prevent an endlessly running transaction and is often an indication that there is an underlying performance issue (for instance a slow-running SQL) or inherent limit on how much work can be accomplished in the time period (such as attempting to generate a bill online for an account with hundreds of contracts, which should be done in batch).

2.9.1 Analytics Publisher Limits

The table below lists several pre-set limits in Analytics Publisher for reporting:

Process/Report	**Limit**
Execution of SQL to build an Online report	10 minutes
Execution of SQL to build a report (Scheduled (offline) report output)	30 minutes
XML format report output	500 MB
CSV format report output	1,000,000 Rows
Online/Browser report output	300 MB
Scheduled (offline) report output	500 MB

2.9.2 Oracle REST Data Services Limits

The table below lists several pre-set limits related to use of Oracle REST Data Services (ORDS), including SQL Developer Web:

Process	**Limit**
SQL Developer Web Online Query Timeout	15 minutes
ORDS REST Query Timeout	15 minutes
ORDS HTTP Timeout	5 minutes (default)

2.9.3 Web Service Limits

The table below lists several pre-set limits related to use of web services:

Process	**Default**	**Limit**
Web Service Call	1 minute	5 or 15 minutes **Note:** The timeout limit for web service calls can be extended to either 5 minutes or 15 minutes if a service request is submitted to the Oracle Revenue Management and Billing Cloud Service Operations team.

Note: Customers and implementers should review any HTTP request that is timing out as this would suggest the underlying service has issues and consider asynchronous methods vs synchronous where appropriate.

2.10 Server Logs - Online and Batch

The Oracle Utilities Application Framework creates log files for various processes such as web server, application server, and batch processes. Since log files may contain both technical and personal private information intended to be accessible by different people, the logs have been split with this technical and "application user" separation in mind.

Application users can view the web and application server user logs online by pressing the "Show User Log" button when running the application with "debug=true" in the URL (see the **Debug Mode** section in *Oracle Utilities Application Framework Administrative Guide*). The batch logs can be downloaded on the **Batch Run Tree** page for each thread via the **Download stdout** and **Download stderr** links.

Authorized administrators can also view tracing and debug logs of other users by pressing the "Show Log" button when running the application with "debug=true" in the URL (see the **Debug Mode** section in *Oracle Utilities Application Framework Administrative Guide*).

The application user logs accessible to the user through the above method may not contain certain internal technical details (for example, table structure, SQL, or other internal code-related logging).

3. Data Fix with Plug-In Driven Batch

Inevitably there are fixes required that cannot be done by users through online tools, either due to the complexity of the issue or the volume of data. This section provides guidelines for an approach to these types of fixes using Plug-In Drive batch processes. It contains the following topics:

- Using Plug-In Driven Batch Processes for Fixing Data Issues
- Plug-In Driven Batch Components
- Sample Solution

3.1 Using Plug-In Driven Batch Processes for Fixing Data Issues

Data fixes can often be performed using a Plug-in Driven batch which affords the following benefits:

1. Development and testing can be done in the development environment without the need for Service Requests.
2. The fix will be applied through the application layer ensuring that it is well validated.
3. Creating a plug-in driven batch is a relatively straight forward process that only requires SQL and scripting knowledge.

For a more in depth look at how a plug-in driven batch can be used to execute a data fix, refer to the **Plug-in Driven Background Processes** section in *Oracle Utilities Application Framework Administrative Guide*.

In some cases, the ability to create plug-in driven batch approach will be constrained - for example you cannot change many status values directly via a Business Object update. In such cases, a service request will need to be logged with the required SQL update statement (and expected results - such as number of rows impacted). See the *Oracle Revenue Management and Billing Cloud Service, Premium Edition Operations Guide* for more information.

3.2 Plug-In Driven Batch Components

For those unfamiliar, a plug-in driven batch is a batch control defined with the Java class com.splwg.base.domain.batch.pluginDriven.PluginDrivenGenericProcess. This class orchestrates the execution of a Select Records plugin spot and a Process Records plug-in spot which mimic the functionality typically found in the Java based batch controls.

To leverage this type of batch for a data fix we will use the select records plugin spot to identify the records that require fixing and use a plug-in script (Groovy is also possible) to perform the fix. This will leverage the following:

1. A batch control (using F1-PDBG as a template)
2. A Select Records plug-in script (algorithm)
3. A Process Records plug-in script (algorithm)

3.3 Sample Solution

This section presents a sample use case and one possible solution addressing the use case using a plug-in driven batch process.

Note: If the plug-in creation requirement is for a Production environment, always develop the code in your development environment first, perform initial testing in that environment with sample data, then migrate the code to your Test environment using Content Migration Assistant (CMA) and perform the testing again on actual customer data, and then repeat the same steps to migrate the code to the Production environment.

Warning: Once data has been deleted and or updated using a plug-in driven batch process, the data cannot be restored. As such, Oracle strongly recommends thorough testing of your plug-in driven batch job.

3.3.1 Use Case

Many devices in Oracle Revenue Management and Billing Cloud Service have been created with an incorrect head end system configured directly on the device itself. Manually updating the devices would take too long so an automatic fix is required. The devices in question can be identified by the device type code of ROM-E-SMARTMTR and a head end system of MV90. The solution is to update each of these devices with a new head end value of L+G (Landis+Gyr).

3.3.1.1 Step One: Create the Batch Control

Creating the batch control begins by duplicating the OUAF delivered batch control F1PDBG - Plug-in Driven Generic Template. This batch control provides the appropriate Java class as well as a set of default batch parameters.

For this solution, we will provide flexibility through a set of three batch job parameters:

- Device Type Code
- Current Head End
- Target Head End

The first two will be used to identify the set of devices that need to be fixed and the last parameter will identify the new head end value that will be applied to each of the devices.

Here are the additional parameters that are to be defined:

Sequence	Parameter Name	Description	Detail Description	Required
10	deviceTypeCode	Device Type Code	Device Type Code of the devices to be updated	Yes
20	currentHeadEnd	Current Head End	Current Head End of the devices to be updated	Yes
30	targetHeadEnd	Target Head End	The new Head End that will be applied to the devices selected	Yes

3.3.1.2 Step Two: Create a Select Records Algorithm

This algorithm should be created by duplicating the sample OUAF algorithm type F1PDB-SR - Select Records by Pre-defined Query and the plug-in script F1-PDBSelRec - Select Records by Pre-defined Query.

No changes are required to the algorithm type other than directing it to the new plug-in script that we have made.

The plug-in script logic will set the batch strategy and key field (code from the OUAF sample) with the addition of logic for taking the batch parameters for device type code and current head end system and setting them to the bind parameters from the query we provide.

There are two key collections in the hard parameters of this plug-in spot:

1. `parm/hard/batchParms/BatchParm`: This contains each of the batch parameters as a name/value pair. You will access this list by using the "Parameter Name" value defined on the batch control. For example, to retrieve the device type code you would retrieve the entry in this list with a name of "deviceTypeCode".
2. `parm/hard/bindVariables/Bind`: Populating this list is the key purpose of this plug-in script. The field name is used to ensure the value is bound with proper data typing, so provide the field name that corresponds with the field the bind variable is being compared with. The name should be the name of the bind in the SQL you are providing. The value will be a value you extracted either from the batch parameters or from an algorithm soft parameter (in this example we are not using soft parameters for bind values).

Here is the sample logic we are using for our solution:

```
move "parm/soft[2]/value" to "parm/hard/batchStrategy";
move "parm/soft[3]/value" to "parm/hard/keyField";
//push the batch parameter for device type code to the bind variable
//the field name reflects the database column that this bind will be compared against
//the batch parameter name is based on the parameter name defined on the batch control
//the bind variable name should match the name in the query.  in this instance we used "F1"
//because an OUAF zone was used to test the query and this way the query didn't need to
change
move 'DEVICE_TYPE_CD' to "parm/hard/bindVariables/+Bind/fieldName";
move 'F1' to "parm/hard/bindVariables/Bind[last()]/name";
move "parm/hard/batchParms/BatchParm[name='deviceTypeCode']/value" to
"parm/hard/bindVariables/Bind[last()]/value";
//push the batch parameter for the current head end to the bind variable "F2"
move 'D1_SPR_CD' to "parm/hard/bindVariables/+Bind/fieldName";
move 'F2' to "parm/hard/bindVariables/Bind[last()]/name";
move "parm/hard/batchParms/BatchParm[name='currentHeadEnd']/value" to
"parm/hard/bindVariables/Bind[last()]/value";
```

Here is a sample of the hard parameter data area from an execution of this logic to give a better idea of the inputs to this script:

```
<root>
   <parm>
      <soft>
         <value>
            select d1_device_id from d1_dvc where device_type_cd = :F1
            and d1_spr_cd = :F2 and d1_device_id between :f1.lowId AND
            :f1.highId order by d1_device_id
         </value>
      </soft>
      <soft>
         <value>THDS</value>
      </soft>
      <soft>
         <value>D1_DEVICE_ID</value>
      </soft>
      <hard>
         <batchControl>
            <id>ZZ-TSTDU</id>
         </batchControl>
         <batchParms>
            <BatchParm>
               <name>deviceTypeCode</name>
               <value>ROM-E-SMART-MTR</value>
            </BatchParm>
            <BatchParm>
               <name>targetHeadEnd</name>
               <value>L+G</value>
            </BatchParm>
            <BatchParm>
               <name>currentHeadEnd</name>
               <value>MV90</value>
            </BatchParm>
         </batchParms>
         <batchRunNumber>2</batchRunNumber>
         <businessDate>2019-08-16</businessDate>
         <isNewRun>false</isNewRun>
         <numOfThreads>5</numOfThreads>
         <batchStrategy>THDS</batchStrategy>
      </hard>
   </parm>
```

```
</root>
```

Lastly an algorithm should be created for the algorithm type with the following parameter values:

Parameter	Sequence	Value	Comments
SQL	1	select d1_device_id from d1_dvc where device_type_cd = :F1 and d1_spr_cd = :F2 and d1_device_id between :f1.lowId AND :f1.highId order by d1_device_id	This is a simple SQL that retrieves all the device IDs within a given ID range that have the input device type code and head end system. The SQL sets up 4 bind variables: • :F1 is the device type code that will be provided in the batch parameters • :F2 is the head end that will be provided in the batch parameters • :f1.lowId is a special parameter that will be injected by the batch control with the thread specific low ID. The plug-in script does not need to worry about this bind variable. • :f1.highId is similar to :f1.lowId except it represents the high ID for the thread The low and high ID are part of the query because of our choice for the next parameter. **Note:** There is a 2000 character limit on algorithm parameters so SQL provided must be succinct.
Batch Category	2	THDS	To take advantage of multithreading we have set the batch strategy to THDS which means it will thread based on a range of a table key. In this instance we will use the device ID and the batch program will evenly divide the possible range of device across the available threads and the selecting of records will be done within each thread. This is no different than our standard batch threading mechanism. If for some reason the SQL to identify the items to fix didn't fall nicely into a master data key, you can use the 'JOBS' strategy which will first select the records and then evenly divide them up across the available threads.

Parameter	Sequence	Value	Comments
Key Field	3	D1_DEVICE_ID	This is identifying which of the returned fields by the query is being used in the threading strategy.

3.3.1.3 Step Three: Create a Process Records Algorithm

For this plug-in spot we will create an entirely new plug-in script that will perform the following high level steps:

1. Retrieve the device ID from the SQL results in the hard parameters
2. Use the device ID and a lite business object to read the device information
3. Retrieve the new head end from the batch parameters
4. Set the new head end to the lite business object and perform an update

There are two key collections within the hard parameters of this plug-in spot:

1. `parm/hard/batchParms/BatchParm`: This contains each of the batch parameters as a name/value pair. You will access this list by using the "Parameter Name" value defined on the batch control. For example, to retrieve the target head end you would retrieve the entry in this list with a name of "targetHeadEnd".
2. `parm/hard/selectedFields`: This contains the results of the query. Each entry here has a name that corresponds to a column in the select clause of your query and a value that contains the data selected.

Here is some sample logic for this solution:

```
//retrieve the device ID from the query results. this logic will receive exactly one device at a time.
//use that device ID to perform a read of the device using a lite BO
// a lite BO is being used to make sure this is performing as quickly as possible by eliminating unnecessary data collections
move "parm/hard/selectedFields/Field[name='D1_DEVICE_ID']/value" to "D1-DeviceDetailsLITE/meterId";
invokeBO 'D1-DeviceDetailsLITE' using "D1-DeviceDetailsLITE" for read;
//retrieve the value for the new head end from the batch parameter list
//use that value to perform an update with the lite BO.
//fastUpdate is being used because there is no further processing and a subsequent read after the update is not required
move "parm/hard/batchParms/BatchParm[name='targetHeadEnd']/value" to "D1-DeviceDetailsLITE/headEndSystem";
invokeBO 'D1-DeviceDetailsLITE' using "D1-DeviceDetailsLITE" for fastUpdate;
```

Here is a sample of the hard parameter data area from an execution of this logic to give a better idea of the inputs you have to this script:

```
<root>
   <parm>
      <hard>
         <batchParms>
            <BatchParm>
               <name>currentHeadEnd</name>
               <value>MV90</value>
            </BatchParm>
            <BatchParm>
               <name>deviceTypeCode</name>
               <value>ROM-E-SMART-MTR</value>
            </BatchParm>
           <BatchParm>
               <name>targetHeadEnd</name>
               <value>L+G</value>
           </BatchParm>
        </batchParms>
         <batchParmsHelper>
               <batchControlId>ZZ-TSTDU</batchControlId>
         </batchParmsHelper>
         <isFirst>true</isFirst>
         <isLast>false</isLast>
         <jobParms>
               <batchCode>ZZ-TSTDU</batchCode>
               <batchNumber>2</batchNumber>
               <businessDate>2019-08-16</businessDate>
               <numOfThreads>5</numOfThreads>
               <threadNumber>1</threadNumber>
         </jobParms>
         <selectedFields>
            <Field>
               <name>D1_DEVICE_ID</name>
               <value>114747438643</value>
           </Field>
        </selectedFields>
     </hard>
   </parm>
</root>
```

Update via DTO

In cases that an update needs to be on a maintenance object that is not business object maintained or an update is specific to a table field and not exposed on any business object schema, the object can be maintained via entity/DTO using Groovy Scripting.

```
10: Step Type: Edit Data
move "xs:date(parm/hard/batchParms/ BatchParm[name='billDate']/value)"
to $billDate;
if ("string($billDate) = $BLANK")
terminate with error(6, 16504);
end-if;
move "string(parm/hard/selectedFields/ Field[name='BILL_ID']/value)"
to $billId;
if ("string($billId) = $BLANK")
terminate;
end-if;
invokeGroovy 'updateUsageDTO';
invokeGroovy 'updateBillEntity';

20: Step Type: Groovy Members
void updateUsageDTO(){
Bill_Id billId = new Bill_Id(getStringScriptVariable('billId'))
Bill bill = billId.getEntity()
Account account = bill.getAccount();
String accountIdStr = account.getId().getTrimmedValue();
move accountIdStr, "ZZGetBSUsage/input/accountId";
PreparedStatementQuery query = createPreparedStatement("""
SELECT FT_ID, SIBLING_ID AS BSEG_ID, USAGE_ID FROM CI_FT FT, C1_USAGE
USG WHERE FT.SA_ID IN (SELECT SA.SA_ID FROM CI_SA SA WHERE ACCT_ID =
:accountId AND EXISTS (SELECT 'X' FROM CI_SA_TYPE SATYPE WHERE
SATYPE.SA_TYPE_CD = SA.SA_TYPE_CD AND SATYPE.CIS_DIVISION =
SA.CIS_DIVISION AND SATYPE.SPECIAL_ROLE_FLG = 'BD'))
AND FT.BILL_ID  = ' '
AND FREEZE_SW = 'Y'
AND ft.sibling_id = usg.bseg_id
AND ft.ft_type_flg IN ( 'BS', 'BX')
""", "Get Usage Bill Segment")
```

```
query.bindString("accountId", accountIdStr, "ACCT_ID");
List<SQLResultRow> usageQueryList = query.list();
if (usageQueryList == null) return
BillSegment_Id nonBdBillSegment = new
BillSegment_Id(getStringScriptVariable('nonBdBillSegmentId'));
for(SQLResultRow iter : usageQueryList){
String usageIdStr = iter.get("USAGE_ID")
Usage usage = new Usage_Id(usageIdStr).getEntity()
//update Bill Segment on Usage via DTO
Usage_DTO usageDTO = usage.getDTO()
usageDTO.setBillSegmentId(nonBdBillSegment)
usage.setDTO(usageDTO)
}

30: Step Type: Groovy Members
public void updateBillEntity() {
Bill_Id billId = new Bill_Id(getStringScriptVariable('billId'))
Bill bill = billId.getEntity()
Date accountingDate = getProcessDateTime().getDate()
BillCompletionInputData billCompletionInputData =
BillCompletionInputData.Factory.newInstance()
billCompletionInputData.setAccountingDate(accountingDate)
billCompletionInputData.setBillDate(getDateScriptVariable('billDate'))
billCompletionInputData.setUnableToCompleteBillAction(UnableToComplete
BillActionLookup.constants.SHOW_ERROR)
bill.complete(billCompletionInputData)
```

3.3.1.4 Step Four: Submitting the Batch Job

The final step of the process is to submit a batch job to perform the update. At this point, there is no difference between this style of batch and any other.

4. Troubleshooting

This section provides troubleshooting guidelines for Oracle Revenue Management and Billing Cloud Services. It contains the following topics:

- Database Monitoring Using SQL Developer Web
- Running and Troubleshooting Batch Processing

4.1 Database Monitoring Using SQL Developer Web

Oracle Cloud Infrastructure uses a variety of technologies to monitor the availability and performance of Oracle Cloud Services and the operation of infrastructure and network components. It has been designed using best practices to monitor the processes, data, and access to all cloud technologies within the tenancy.

In addition to comprehensive Oracle Cloud hardware monitoring, Oracle Revenue Management and Billing Cloud Services are designed based on modern best practices to provide service specific monitoring capabilities extending from the tenancies on which the services are housed to monitoring batch, state, and overall performance.

An important aspect of Software-as-a-Service (SaaS) is that it is monitored by Oracle to ensure that the various services in each environment are available (if not, to resolve the problem and take preventive action), however there are several external tools available to the customers for monitoring as well as for performance and tuning.

4.1.1 Performance Hub Report

The DBMS_PERF.REPORT_PERFHUB package/function can be invoked via SQL Developer web (provided with Oracle REST Data Services, or ORDS) to generate a composite active performance hub report of the database system for a specified time period.

This self-service tool generates a performance hub report of the database which can be used by utility/partner to review and analyze the top SQLs, wait events, CPU usage etc.

SQL Developer web (ORDS) displays an explain plan (before execution) which shows the sequence of operations Oracle performs to run the statement. The AutoTrace functionality provides further statistics of the SQL statement (after running the statement) for analysis and tuning.

Use this report when:

- Batch processing is running slow.
- Application portals and zones take a longer than expected time to execute queries.
- Application performance is slow.

This report is also a handy tool for custom development and QA.

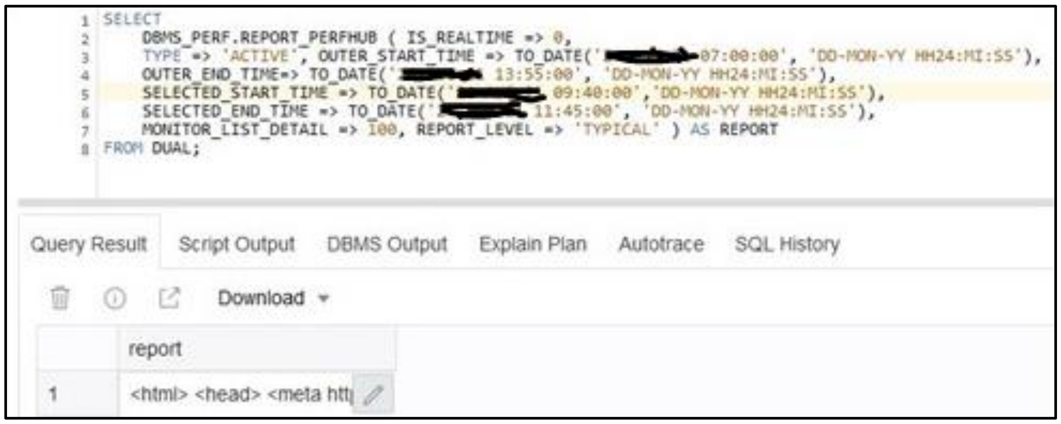
Running the Performance Hub Report

Use the following procedure to generate the Performance Hub Report:

1. Using SQL Developer web (ORDS), run the following sample SQL statement:

   ```
   SELECT DBMS_PERF.REPORT_PERFHUB (IS_REALTIME => 0, TYPE =>
   'ACTIVE', OUTER_START_TIME => TO_DATE('DD-MM-YY 07:00:00', 'DD-
   MON-YY HH24:MI:SS'), OUTER_END_TIME=> TO_DATE('DD-MON-YY
   13:55:00', 'DD-MON-YY HH24:MI:SS'), SELECTED_START_TIME =>
   TO_DATE(''DD-MON-YY09:40:00','DD-MON-YY HH24:MI:SS'),
   SELECTED_END_TIME => TO_DATE(''DD-MON-YY11:45:00', 'DD-MON-YY
   HH24:MI:SS'), MONITOR_LIST_DETAIL => 100, REPORT_LEVEL =>
   'TYPICAL' ) AS REPORT FROM DUAL;
   ```

   The following illustrates this SQL statement in SQL Developer web:



```
SELECT
    DBMS_PERF.REPORT_PERFHUB ( IS_REALTIME => 0,
    TYPE => 'ACTIVE', OUTER_START_TIME => TO_DATE('[redacted]07:00:00', 'DD-MON-YY HH24:MI:SS'),
    OUTER_END_TIME=> TO_DATE('[redacted] 13:55:00', 'DD-MON-YY HH24:MI:SS'),
    SELECTED_START_TIME => TO_DATE('[redacted] 09:40:00','DD-MON-YY HH24:MI:SS'),
    SELECTED_END_TIME => TO_DATE('[redacted] 11:45:00', 'DD-MON-YY HH24:MI:SS'),
    MONITOR_LIST_DETAIL => 100, REPORT_LEVEL => 'TYPICAL' ) AS REPORT
FROM DUAL;
```

	report
1	<html> <head> <meta htt

Figure 1: SQL Developer Web

2. Export the output in xml format to the local file system and then change the file extension to .html.
3. Open the file in any browser and review the list of SQLs executed over a period of time, including wait events, CPU usage, and so on.

Performance Hub Report Parameters

The following table lists parameters that can be used when running the Performance Hub Report:

Parameter	Description
is_realtime	If 1, then real-time. If NULL (default) or 0, then historical mode.
outer_start_time	Start time of outer period shown in the time selector. If NULL (default): • If is_realtime=0 (historical), then 24 hours before outer_end_time. • If is_realtime=1 (realtime mode), then 1 hour before outer_end_time.

Parameter	Description
outer_end_time	End time of outer period shown in the time selector. If NULL (default), then latest AWR snapshot. If is_realtime=0 (historical), then the latest AWR snapshot. If is_realtime=1 (realtime mode), this is the current time (and any input is ignored).
selected_start_time	Start time period of selection. If NULL (default): • If is_realtime=0, then 1 hour before selected_end_time. • If is_realtime=1, then 5 minutes before selected_end_time.
selected_end_time	End time period of selection. If NULL (default): • If is_realtime=0, then latest AWR snapshot. • If is_realtime=1, then current time.
inst_id	Instance ID for which to retrieve data: • If -1, then current instance. • If number is specified, then for that instance. • If NULL (default), then all instances.
dbid	DBID to query: • If NULL, then current DBID. • If is_realtime=1, then DBID must be the local DBID.
monitor_list_detail	Top N in SQL monitor list for which to retrieve SQL monitor details: • If NULL (default), then retrieves top 10. • If 0, then retrieves no monitor list details.
workload_sql_detai	Top N in Workload Top SQL list to retrieve monitor details: • If NULL (default), then retrieves top 10. • If 0, then retrieves no monitor list details.
addm_task_detail	Maximum N latest ADDM tasks to retrieve: • If NULL (default), retrieves available data but no more than N. • If 0, then retrieves no ADDM task details.
report_reference	Must be NULL when used from SQL*Plus.
report_level	'typical' will get all tabs in performance hub.
type	Report type: • 'ACTIVE' (default) • 'xml' returns XML

Parameter	Description
base_path	URL path for HTML resources since flex HTML requires access to external files. This is only valid for type='ACTIVE' and is typically not used. Default value will retrieve the required files from OTN.

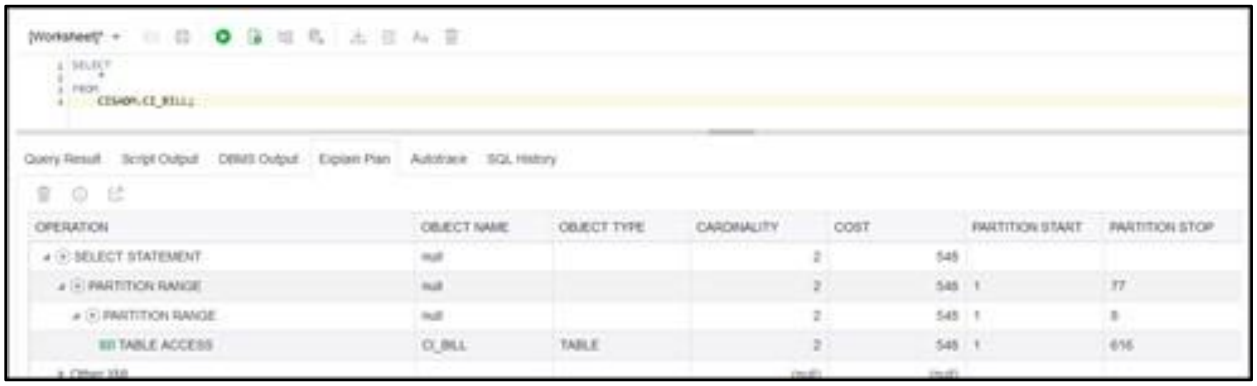
Important Notes:

- The Performance Hub Report provides details based on the specified date range parameter. In absence of date range parameter, report will provide data based on the last 15 minutes.
- The target audience of this tool are the local database administrators assigned monitoring and troubleshooting responsibilities.
- The AutoTace functionality follows the ORDS timeout guideline.
- While raising performance related issues via a service request, the customer is required to attach the performance hub report.
- For more details on Performance Hub Report, please visit online documentation (https://docs.oracle.com/en/database/oracle/oracle-database/index.html).

Reviewing and Tuning SQL Statements

SQL statements retrieved using the Performance Hub Report can be reviewed and tuned using the SQL Developer web (ORDS) Explain Plan and AutoTrace functionalities.

The following figure provides a sample of the Explain Plan function.

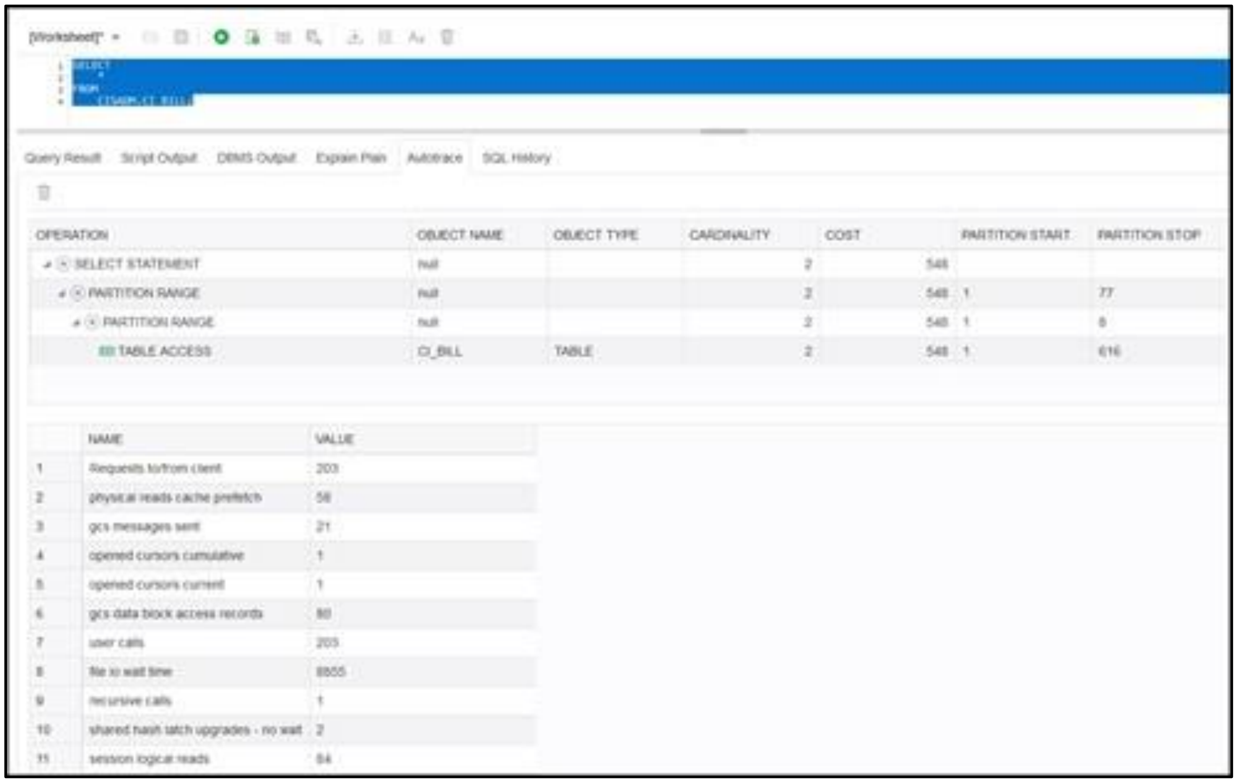


```
SELECT
    *
FROM
    CISADM.CI_BILL;
```

OPERATION	OBJECT NAME	OBJECT TYPE	CARDINALITY	COST	PARTITION START	PARTITION STOP
SELECT STATEMENT	null		2	545		
PARTITION RANGE	null		2	545	1	77
PARTITION RANGE	null		2	545	1	8
TABLE ACCESS	CI_BILL	TABLE	2	545	1	616
Other XML			(null)	(null)		

Figure 2: Explain Plan

The following figure provides a sample of the AutoTrace function.



```
SELECT
    *
FROM
    CISADM.CI_BILL
```

OPERATION	OBJECT NAME	OBJECT TYPE	CARDINALITY	COST	PARTITION START	PARTITION STOP
SELECT STATEMENT	null		2	548		
PARTITION RANGE	null		2	548	1	77
PARTITION RANGE	null		2	548	1	8
TABLE ACCESS	CI_BILL	TABLE	2	548	1	616

	NAME	VALUE
1	Requests to/from client	203
2	physical reads cache prefetch	58
3	gcs messages sent	21
4	opened cursors cumulative	1
5	opened cursors current	1
6	gcs data block access records	80
7	user calls	205
8	file io wait time	8855
9	recursive calls	1
10	shared hash latch upgrades - no wait	2
11	session logical reads	84

Figure 3: AutoTrace

4.2 Running and Troubleshooting Batch Processing

This section provides guidelines for running and troubleshooting batch processing with Oracle Revenue Management and Billing Cloud Service. It contains the following topics:

- Introduction
- Batch Basics
- Important Parameters
- Batch Performance Factors
- Frequently Asked Questions for Common Batch Issues

4.2.1 Introduction

Batch jobs are used for several key purposes in Oracle Revenue Management and Billing Cloud Services - to load data from files, to process new data that has been loaded/entered, to monitor status of various objects, etc. Batch jobs can be run individually on an ad hoc basis, or as part of a scheduled batch job stream with dependencies. The system produces logs during batch run execution, and it's important to understand how to view and interpret these logs.

Especially during implementation there are some common problems that may crop up during execution of jobs and streams. The following guide is designed to help you understand how to run batch and troubleshoot batch job problems and describe important triage steps to perform before logging a Service Request (SR) for a batch problem.

4.2.2 Batch Basics

The system provides multi-threaded batch processing capability for many jobs. To support this, every cloud service environment has two Threadpool Workers (TPW) - Default and NOCACHE. Most jobs should be run with the Default TPW, but there are a few that need to use NOCACHE (Jobs related to CMA for configuration migration, jobs like K1-RIUSP that are building database structures and require longer than normal time for single operations). You have the capability to specify how many threads from the TPW should be assigned to your job submission. There is a capability for a user to 'Restart TPWs' in case of a severe problem where jobs are not getting started - this can happen especially during implementation when data quality is variable.

Each time you run a particular batch, it gets a unique run number and has a status (Pending, In Progress, Complete, Error), and has one or more threads (each thread has a status as well, Pending, Complete, Error). A thread can have multiple instances if for instance the first instance errored out, and you restarted the job. Sometimes you will want to ensure that no more instances are created for a run, if for example you need to reset the parameters of the job. Details of each batch run are displayed on the Batch Run Tree.

The service also provides the capability to define batch job streams - a linked sequence of batch jobs to run either on a pre-set schedule or ad hoc, with ability to set dependencies - i.e., only start job C after both job A and job B have completed successfully. A stream may fail to complete if one of the jobs has a problem - so if that happens you need to investigate the outcome of the job.

Note that this guide assumes familiarity with the Oracle Utilities Application Framework batch framework. The online documentation can provide much more detail on all the online transactions related to batch processing.

4.2.3 Important Parameters

The system provides many batch jobs, and they are defined in the Batch Control table. There are common parameters applicable for all batch jobs, and most jobs also have unique parameters as well.

- **Override Number of Records to Commit**: We recommend not overriding to a number greater than 1 (i.e., don't override to a bigger number thinking that will help - the infrastructure works best to commit frequently).
- **Thread Pool Name**: Parameter used to define the name of the thread pool worker used with the batch process. Valid values are DEFAULT and NOCACHE. As stated above, be aware of the jobs that should be run with NOCACHE TPW.
    - NOCACHE TPW should **only** be used with Conversion, Content Migration Assistant (CMA) related batch jobs such as K1-RIUSP, , F1-MGOAP, F1-MGTAP, and so on.
    - NOCACHE TPW should **not** be used for batch jobs related to Generalized/Specialized Data Export or any other functional batch jobs.

  Leave the **Thread Pool Name** parameter blank to use the DEFAULT TPW.

4.2.4 Batch Performance Factors

Batch performance is dependent on several factors, and these factors can be changing rapidly during implementation, so it's important to have a mental checklist to review as batch processing is being tested:

State of data conversion - typically data conversion involves multiple iterations, and data quality will vary (hopefully increasing steadily!). We strongly recommend use of the validation batch programs to do statistical sampling of the loaded data, to help identify problems such as incorrect or missing foreign keys - you can review the results on the **FK Validation Summary** and **Validation Error Summary** portals. If there are data quality issues, they will often result in errors during various types of batch processing. Be aware of what's happening with data conversion.

- State of indexes and the database - during implementation and data conversion iterations, frequently tables are being truncated and re-loaded, and indexes may be disabled then re-built. Problems will occur if any indexes get into 'unusable' state. Ensure the data conversion team always follows necessary steps and gives official go-ahead before running your jobs in an environment where data conversion is active. To check for unusable indexes, you can run this query: SELECT INDEX_NAME, TABLE_NAME FROM all_indexes where TABLE_OWNER = 'CISADM' and STATUS = 'UNUSABLE' ORDER BY TABLE_NAME.
- Extensions - new scripts and algorithms may have unexpected impacts on batch processing if they have not been fully designed and tested for scalability. Best practice is to review the explain plans for all new SQL used in extensions, to ensure that new code will perform suitably.
- Other concurrent processing - during implementation often various teams are using one environment and trying various jobs. Be aware of what other activities may be happening concurrently - those activities may impact your job in some way.
- Multi-threading setup - while many jobs are designed to be run multi-threaded (to divide the workload into roughly equivalent groupings to process in parallel), the performance characteristics are not necessarily linear. By this we mean that doubling the number of threads for a job will not always cut the processing time in half. In general, the best strategy is to start with small data volumes and fewer threads to establish a baseline of data quality and performance, then increase in incremental steps to tune performance.

4.2.5 Frequently Asked Questions for Common Batch Issues

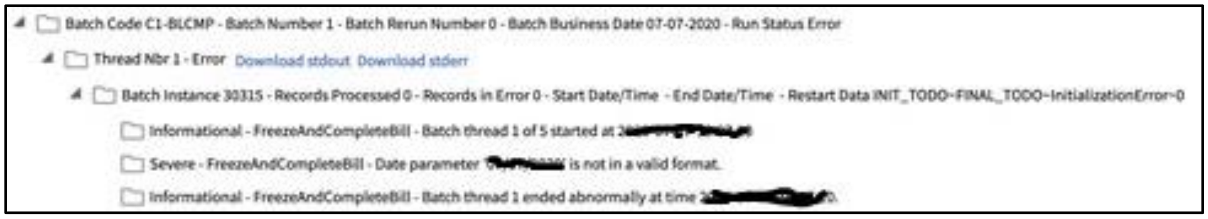
This section lists some of the frequently asked questions with respect to common batch issues.

4.2.5.1 BATCH JOB RESULTS IN ERROR, WHAT SHOULD I DO?

First step is review the Batch Run Tree. The Batch Run Tree shows the details for the Run, Instances and Threads, and is where you can access batch logs (stdout and stderr).

Review the status of the threads by expanding the tree fully. In some cases, the thread details will show the error message that was thrown and that may be enough to understand the problem.

Example 1:



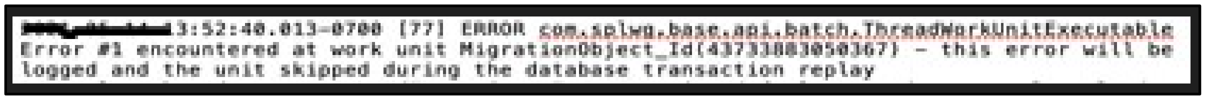
```
Batch Code C1-BLCMP - Batch Number 1 - Batch Rerun Number 0 - Batch Business Date 07-07-2020 - Run Status Error
  Thread Nbr 1 - Error  Download stdout  Download stderr
    Batch Instance 30315 - Records Processed 0 - Records in Error 0 - Start Date/Time - End Date/Time - Restart Data INIT_TODO~FINAL_TODO~InitializationError~0
      Informational - FreezeAndCompleteBill - Batch thread 1 of 5 started at [redacted]
      Severe - FreezeAndCompleteBill - Date parameter [redacted] is not in a valid format.
      Informational - FreezeAndCompleteBill - Batch thread 1 ended abnormally at time [redacted].
```

Figure 4: Batch Run Tree

In the example above, the run is in error, the thread is in error, and the error message is shown clearly under the instance. In this case the problem was with a batch date parameter.

To get more details, download the log files by clicking on the links and search for terms such as 'error ' or 'ORA-' (for database error code).

Example 2:



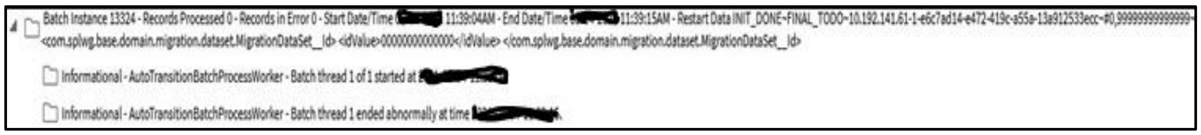
```
[redacted]3:52:40.013-0700 [77] ERROR com.splwg.base.api.batch.ThreadWorkUnitExecutable
Error #1 encountered at work unit MigrationObject_Id(43733883050367) - this error will be
logged and the unit skipped during the database transaction replay
```

Figure 5: Error Batch Log

Above is an example of an error in a batch log - in this case a problem during a CMA job. Note that here the key for a Migration Object is given and is likely a good place to look for something anomalous.

To get more details, download the log files.

Example 3:



```
Batch Instance 13324 - Records Processed 0 - Records in Error 0 - Start Date/Time [redacted] 11:39:04AM - End Date/Time [redacted] 11:39:15AM - Restart Data INIT_DONE~FINAL_TODO~10.192.141.61-1-e6c7ad14-e472-419c-a55a-13a912533ecc~#0,99999999999999~
<com.splwg.base.domain.migration.dataset.MigrationDataSet_Id> <idValue>0000000000000</idValue> </com.splwg.base.domain.migration.dataset.MigrationDataSet_Id>
  Informational - AutoTransitionBatchProcessWorker - Batch thread 1 of 1 started at [redacted]
  Informational - AutoTransitionBatchProcessWorker - Batch thread 1 ended abnormally at time [redacted].
```

Figure 6: Run Tree Error

Here the run tree shows that the thread ended abnormally. Stderrr shows:

```
ERROR org.hibernate.engine.jdbc.spi.SqlExceptionHelper Exception
occurred while getting connection:
oracle.ucp.UniversalConnectionPoolException: Cannot get Connection
from Datasource: java.sql.SQLRecoverableException: Listener refused
the connection with the following error: ORA-12514, TNS:listener does
not currently know of service requested in connect descriptor
```

The above example illustrates a connectivity issue. In cases like this, please raise a SR and provide screenshots of batch job submission with parameters, batch run tree (expanded) and both logs (STDERR and STDOUT).

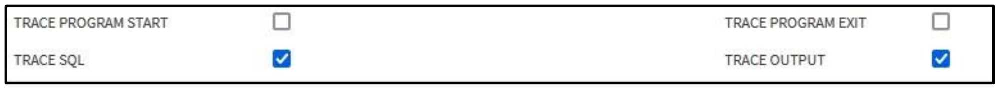
For Batch logs, Oracle maintains logs data for 14 days only, so we recommend that you download them promptly.

Please note that a batch run may complete successfully even though certain records may have encountered recoverable errors (and in certain cases may create To Do entries for follow-up). So particularly in early batch testing it is important to examine the batch run tree for each job to verify the execution details - it's not necessarily enough to just see that the batch completed.

Important Tips:

If the logs are not pointing towards the exact error, logs are emptied or logs don't guide , please follow the steps below:

- Make sure the past runs of batch are marked as DO NOT ATTEMPT RESTART on the Batch Run Tree, Run Control tab.
- Re-submit the batch job with single thread and check the following two boxes on the batch job submission.



TRACE PROGRAM START	☐	TRACE PROGRAM EXIT	☐
TRACE SQL	✓	TRACE OUTPUT	✓

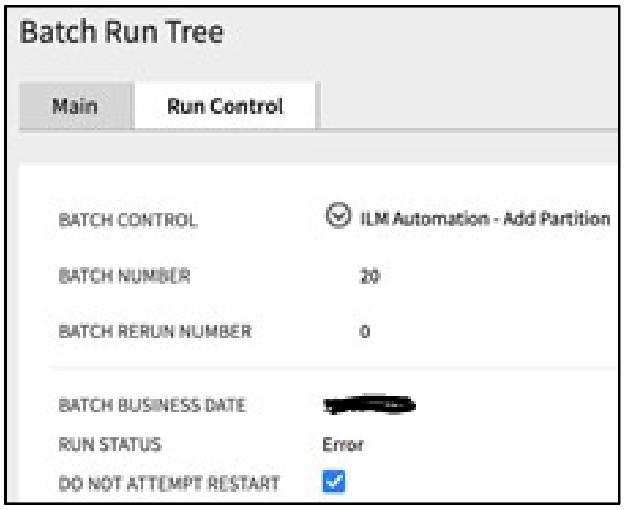
Figure 7: Batch Job Submission

- Once the batch job results in error, go to batch run tree and download and review the logs.
- Make sure to populate MAX-ERRORS/maxErrors batch job parameter (where applicable) with the small number like 10. This parameter will abort the batch run after the batch encounters 10 errors. You may increase this number as per your need, if required.
- DO NOT run the CMA batch jobs with all traces ON, unless it is deemed necessary.
- If there is a need to raise a Service Request, please provide screenshots of the output from the Prepare Issue Details feature (see the **Prepare Issue Details** section in the *Oracle Utilities Application Framework Administrative Guide*), batch job submission with parameters, batch run tree (expanded), and both logs (STDERR and STDOUT).

Note: If the errored batch is custom-built, please contact the project implementation/support team.

4.2.5.2 AFTER A BATCH RESULTS IN ERROR AND AFTER RE-SUBMISSION, WHY DOES THE BATCH RUN TREE SHOWS THE SAME BATCH NUMBER AND BATCH RUN TREE SHOWING MULTIPLE BATCH INSTANCES?

By default, a re-submission of an errored job will cause the previously errored run to restart. Sometimes that is not what you would like to have happen, so to prevent a restart and actually create a new batch run, you need to go to the Run Control tab page of last Batch Run Tree and check the 'Do Not Attempt Re-Start' box (example below).



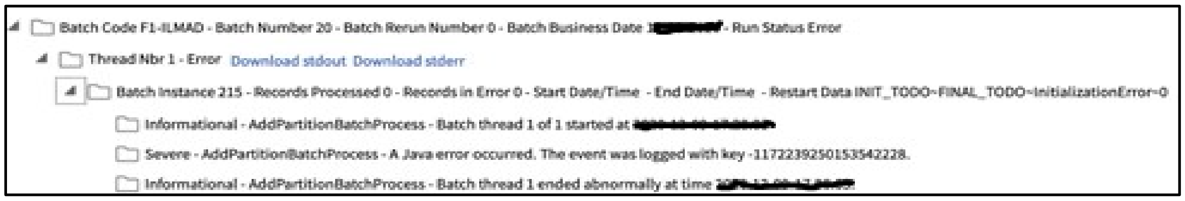
BATCH CONTROL	ILM Automation - Add Partition
BATCH NUMBER	20
BATCH RERUN NUMBER	0
BATCH BUSINESS DATE	[redacted]
RUN STATUS	Error
DO NOT ATTEMPT RESTART	✓

Figure 8: Batch Run Tree

Above is an example where the 'Do Not Attempt Restart' has been checked - this will ensure the next job submission will create a new Batch Number.

4.2.5.3 THE BATCH RUN TREE SHOWS THAT THERE WAS A 'SEVERE JAVA ERROR' - HOW CAN I GET FURTHER DETAILS?

In this case, even the Batch logs may not provide the detail - most often the underlying case is a data problem, such as a bad or missing foreign key that the job is trying to access, causing a null pointer exception. If so, you will need to log an SR and Dev Ops will need to review technical logs to find the underlying error.



```
Batch Code F1-ILMAD - Batch Number 20 - Batch Rerun Number 0 - Batch Business Date [redacted] - Run Status Error
  Thread Nbr 1 - Error  Download stdout  Download stderr
    Batch Instance 215 - Records Processed 0 - Records in Error 0 - Start Date/Time  - End Date/Time  - Restart Data INIT_TODO~FINAL_TODO~InitializationError~0
      Informational - AddPartitionBatchProcess - Batch thread 1 of 1 started at [redacted]
      Severe - AddPartitionBatchProcess - A Java error occurred. The event was logged with key -1172239250153542228.
      Informational - AddPartitionBatchProcess - Batch thread 1 ended abnormally at time [redacted]
```

Figure 9: Severe Java Error

Above is a sample Batch Run Tree with a Java error, please follow the 'Important TIPS' for troubleshooting shown under Question 1.

4.2.5.4 I HAVE SUBMITTED BATCH ONLINE AND THE BATCH JOB SUBMISSION STILL SHOWS BATCH STATUS IN PENDING?

There could be several reasons for this situation:

- All available threads in the TPW are currently busy with another job. To check for this, go to the Batch Queues page and verify if any running jobs.
- The TPW may have been (or need to be) restarted because of a previous problem. If already Restarted, your job should start within minutes. Note that the Restart action takes a bit of time because it may need to interrupt running jobs, cleanly shut the batch server pods down, then start them up again. To restart, please run the application in debug mode where TPW restart option is available (as seen below).

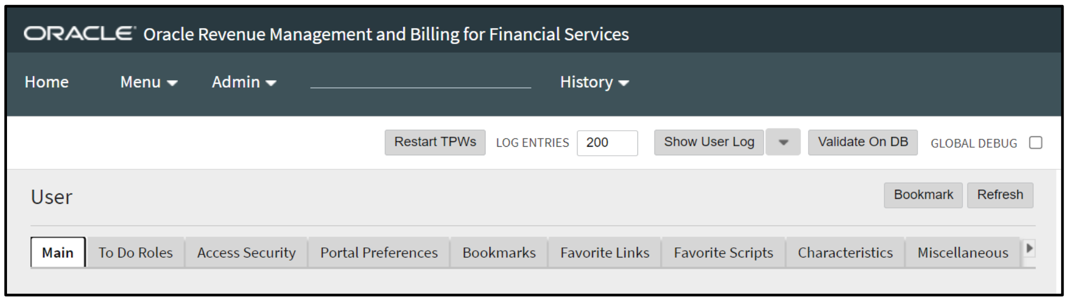


Figure 10: Batch Queue

- Check the desired execution date/time, it should not be in future.

4.2.5.5 I HAVE CANCELLED THE ONLINE BATCH SUBMISSION, BUT BATCH STATUS STILL SHOWS AS PENDING CANCEL?

When you cancel the running batch job with multiple threads, the batch job tries to revert the work done or in progress after the last commit executed. In some cases, where batch job ran with many threads, it will take 4-5 minutes to complete the background process and change the status from pending cancel to canceled. During the process, it is recommended that implementation should not go immediately to restart the TPW. Even if the status don't change after TPW restart, please raise an SR for cloud operations help.

4.2.5.6 I WANT TO SUBMIT A BATCH, BUT I DON'T KNOW WHAT BATCHES ARE CURRENTLY RUNNING?

To check this, go to the **Admin → B → Batch Queues** Portal page and verify if any running jobs. If you see batch jobs already showing in pending status that means you have consumed all the available threads allocated and no capacity available to submit any more batches.

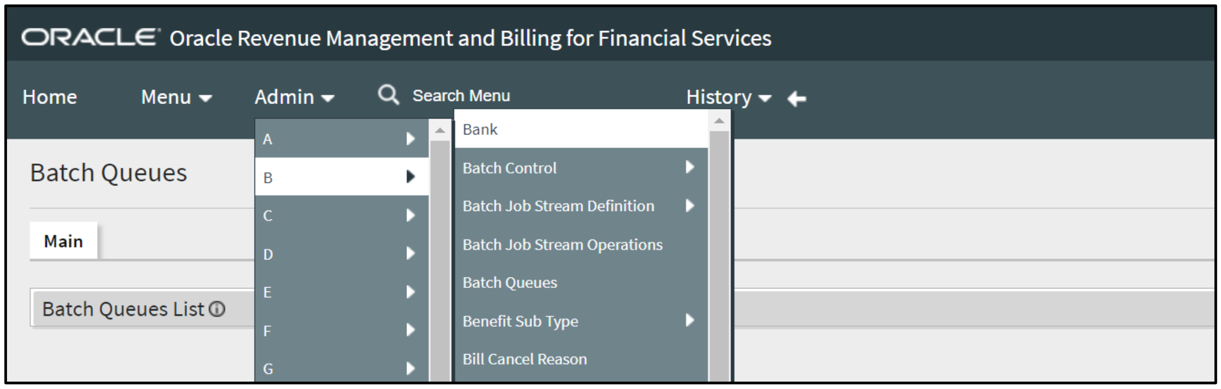
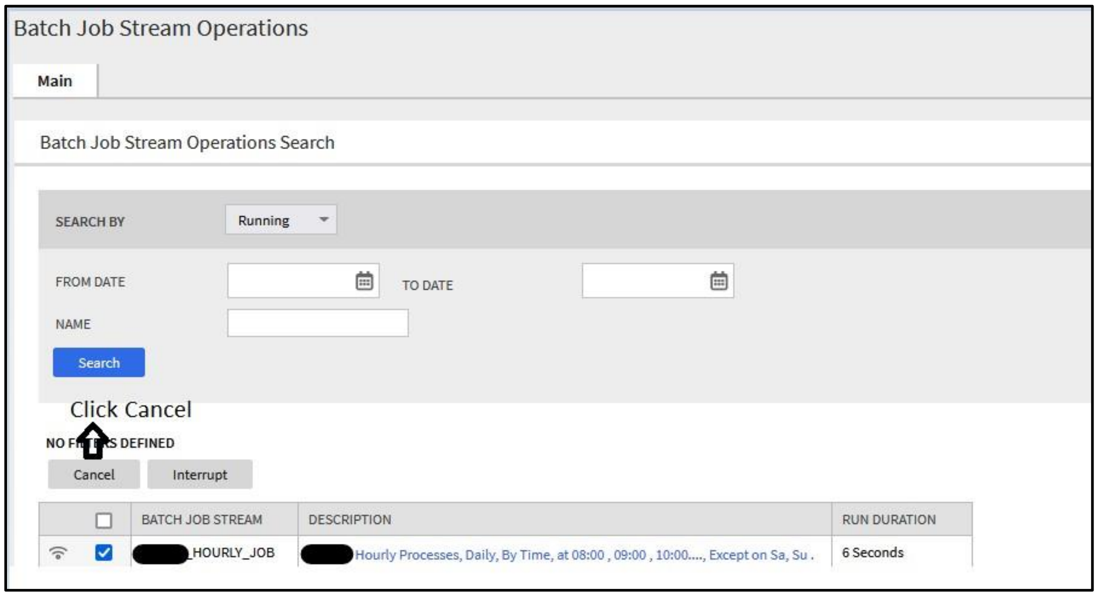


Figure 11: Batch Queues

4.2.5.7 I AM TRYING TO CANCEL THE BATCH SUBMITTED BY DBMS SCHEDULER, IT IS GIVING ME ERROR. HOW TO CANCEL JOBS SUBMITTED BY DBMS SCHEDULER?

A Batch Job Stream executes the batch controls in defined chronological order, and it creates batch job submissions with the submission method "Scheduled". Batch job submission with this status cannot be cancelled through the cancel option. To cancel such batches, you must go to **Admin → B → Batch Job Stream Operations** Portal page, then select the running stream and perform cancel operation. This will result in canceling the overall stream and all the current running batches in that stream.


Batch Job Stream Operations

Main

Batch Job Stream Operations Search

SEARCH BY: Running

FROM DATE | TO DATE

NAME

Search

Click Cancel

NO FILTERS DEFINED

Cancel | Interrupt

	BATCH JOB STREAM	DESCRIPTION	RUN DURATION
✓	[redacted]_HOURLY_JOB	[redacted] Hourly Processes, Daily, By Time, at 08:00 , 09:00 , 10:00...., Except on Sa, Su .	6 Seconds

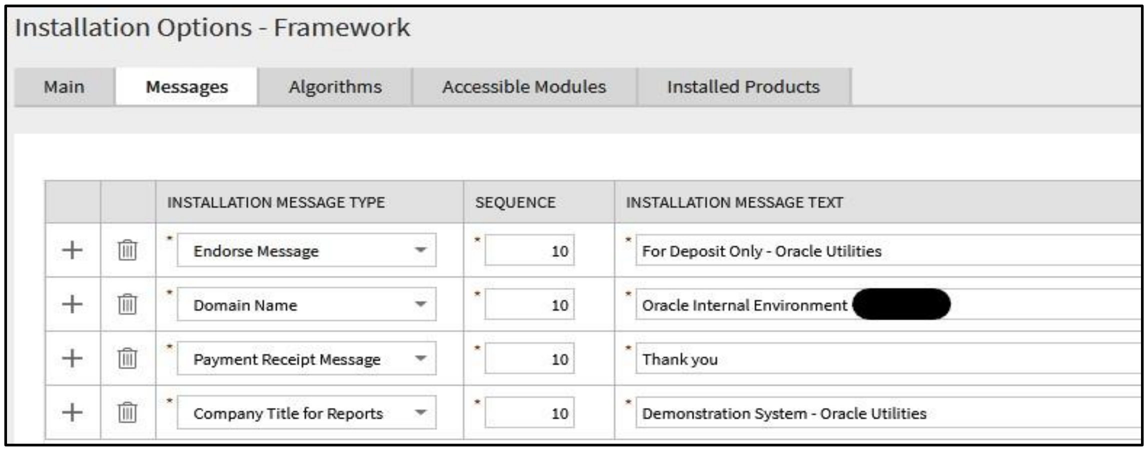
Figure 12: Batch Jobs Stream Operations

4.2.5.8 WHY DO I SEE MULTIPLE INSTANCES OF F1-BTMON BATCH IN BATCH JOB SUBMISSION?

F1-BTMON is the batch probe - which is run automatically as part of the cloud service monitoring of each environment, to verify that batch can execute. This job does a small amount of transitory database access, and if it were to fail alerts would be created for the Cloud Operations team to investigate. It does not use any of the Default or NOCACHE threads so has no impact on your workloads.

4.2.5.9 I AM GETTING BATCH COMPLETION EMAILS FROM MULTIPLE ENVIRONMENTS; CAN I GET ENVIRONMENT NAME INSIDE THE EMAIL CONTENTS?

Yes, from 21A onward, implementations can add a "Domain Name" Installation Message type (on the **Messages** tab of the **Installation Options - Framework** portal) and provide the description that identifies the environment. That description will be part of the email received as part of batch completion as well as on the application interface.


Installation Options - Framework

Main | Messages | Algorithms | Accessible Modules | Installed Products

INSTALLATION MESSAGE TYPE	SEQUENCE	INSTALLATION MESSAGE TEXT
Endorse Message	10	For Deposit Only - Oracle Utilities
Domain Name	10	Oracle Internal Environment [redacted]
Payment Receipt Message	10	Thank you
Company Title for Reports	10	Demonstration System - Oracle Utilities

Figure 13: Messages Tab

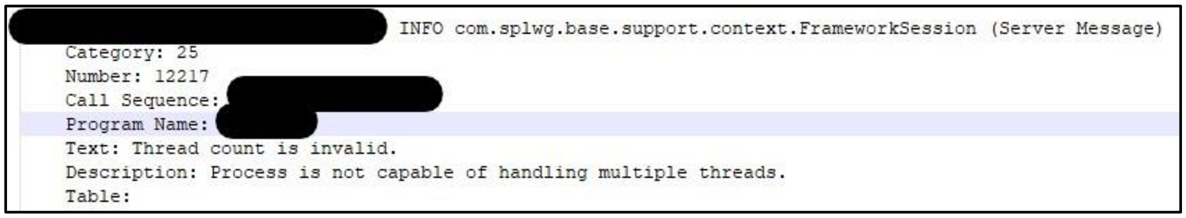
Above is the Installation Options - Framework list of Messages - add a unique value for "Domain Name" in each environment where batch is running.

4.2.5.10 CAN I RUN ANY JOB WITH MANY THREADS?

No, not all jobs support multi-threading. Most Batch Controls will indicate if they support it or not. If the work to be done can be cleanly divided, then multi-threading is available. There are some jobs that take a 'set-based' approach which need to run with a single thread.

If the batch control does not support multi-thread and related to upload/download and you submit the batch with more than one thread, only one thread will process, rest of the threads will start and complete immediately without process any record.

If the batch control does not support multi-thread and not related upload/download, you may see the following error in the log:



```
[redacted] INFO com.splwg.base.support.context.FrameworkSession (Server Message)
    Category: 25
    Number: 12217
    Call Sequence: [redacted]
    Program Name: [redacted]
    Text: Thread count is invalid.
    Description: Process is not capable of handling multiple threads.
    Table:
```

Figure 14: Error Log

4.2.5.11 BATCH JOB SUBMISSION SHOWS 'STARTED' BUT THE BATCH RUN TREE SHOWS STATUS AS PENDING. WHAT IS WRONG?

Typically, this is a temporary situation. The reason for the delay is that in a multi-threaded submission the first thing that is done is the division of work across all the threads via an 'outer select' to get all eligible records - the threads can't get started until the 'work assignments' are complete.

4.2.5.12 I HAVE A JOB THAT I'M RUNNING WITH MANY THREADS, AND ALL THREADS SHOW 'COMPLETE' EXCEPT ONE WHICH APPEARS TO BE 'STUCK' - WHAT CAN I DO?

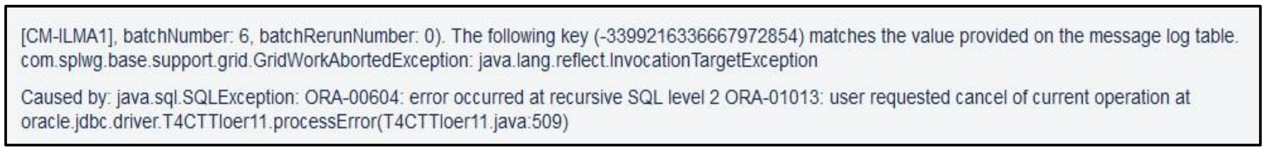
We have seen a couple reasons for this situation - sometimes it really is not 'stuck' - but it is doing a large amount of work on a single record (for instance billing an Account with many contracts). If the record count on the thread remains at the same number for more than 5 minutes, you may have to restart the TPW and or cancel the running batch. After restart, you may submit the batch job again without checking the 'do not attempt to restart' option on the last batch run tree, to restart against what's left to process.

4.2.5.13 THE LOGS FOR MY BATCH JOB ARE EMPTY - WHAT SHOULD I DO?

There are a handful of jobs that will not produce logs, all jobs should have logs, and if logs are empty, first try to submit the batch job with single thread with trace on (please refer to 'Important TIPS' stated above), if you still don’t see the logs, please raise an SR for it, indicating the standard information around the environment, the job, the time it was run. Note that batch logs are only kept for 14 days.

4.2.5.14 THE LOG SAYS THAT THE USER REQUESTED CANCELLATION - BUT WE DIDN'T DO ANY CANCEL - WHAT DOES IT MEAN?

This error typically indicates that a timeout happened during the batch execution. This can happen if one of the jobs (Conversion) that should run in NOCACHE is run in Default by mistake. (For instance, the re-build of an index on a large table can take some time, and under the Default TPW the operation might time out). Time out for Default TPW is 15 mins.



```
[CM-ILMA1], batchNumber: 6, batchRerunNumber: 0). The following key (-3399216336667972854) matches the value provided on the message log table.
com.splwg.base.support.grid.GridWorkAbortedException: java.lang.reflect.InvocationTargetException

Caused by: java.sql.SQLException: ORA-00604: error occurred at recursive SQL level 2 ORA-01013: user requested cancel of current operation at
oracle.jdbc.driver.T4CTTIoer11.processError(T4CTTIoer11.java:509)
```

Figure 15: Batch Log

The above sample log entry typically indicates a timeout issue.

4.2.5.15 THE RECORD COUNTS ON THE THREADS DON'T SEEM TO MATCH UP WITH WHAT I WAS EXPECTING - WHAT IS BEING COUNTED?

Many batch jobs do provide record counts, but typically what is being counted is the unit of work that is driving the process, not the new objects resulting from the processing.

So, for example Billing job will count the number of eligible Accounts to bill, not the number of bills generated.

4.2.5.16 HIGH VOLUME PROCESSING - HOW TO TROUBLESHOOT?

As mentioned in the batch performance factors, the recommended approach to batch is to start small, establish some baselines in terms of records per second or execution time, and then scale up. If you immediate jump to trying a job with hundreds of threads, you may actually be encountering multiple problems at the same time (data quality, threading, etc.) and it can be harder to untangle. If you do start having problems with a job that has successfully run with many threads, then the key is to do what you can to isolate the problem you are hitting by scaling back down again temporarily so you can fully track the variables and changes.

On a batch job submission there are four tracing options that you can turn on for a particular run during testing - these add much more detail to the batch logs, and should be used sparingly (i.e., if you turn on tracing for a job with hundreds of threads, the logging is so voluminous that the log files themselves become hard to manage). When needed, turn on the tracing for a small sample batch run with a single thread, to keep the output manageable.

4.2.5.17 WHILE RUNNING PLUG-IN DRIVEN BATCH FOR DML OPERATIONS ON ADMIN TABLES, I AM GETTING "ILLEGAL ATTEMPT TO MODIFY READ-ONLY ENTITY" ERROR. HOW TO ADDRESS IT?

Run the batch in "NOCACHE" TPW and it will resolve the issue.

In general terms, ‘DEFAULT’ TPW caches a whole lot of ‘Admin’ data including metadata. This makes jobs like billing run a lot faster, because you can safely assume that the admin data won’t be changed during the run, and you can just access what’s in the cache and therefore, If your job does try to make changes to Admin data, you will get errors when running it in "DEFAULT" TPW.

The situation is different when running CMA batch jobs which try to add / update / delete Admin data which need "NOCACHE" (with L2 cache off) that will enable CMA batch to make changes to Admin data while running the batches.

The timeout of NOCACHE has been lifted to cater for longer-running processes (like indexing and partitioning). DEFAULT has a shorter timeout, because in normal jobs it’s a bad sign if operations are taking 15 minutes (or whatever the timeout is set to).

In the meantime, you can of course make changes to Admin data – but sometimes you must do the ‘flush’ to empty and rebuild certain caches so that the new/changed values are accessible to everyone.