Oracle Utilities Cloud Services

Live Operations Guide

For 22B Releases

F58599-03

August 2022
(Revised September 2022)

ORACLE

Oracle Utilities Cloud Services 22B Live Operations Guide

Contents

Chapter 7

Chapter 1

Cloud Services Live Operations Guide

Welcome to the Oracle Utilities Cloud Service Live Operations Guide. This guide includes information regarding live operation of the following Oracle Utilities Cloud Services:

- Oracle Utilities Billing Cloud Service
- Oracle Utilities Customer Care and Billing Cloud Service
- Oracle Utilities Customer Cloud Service
- Oracle Utilities Generation Asset Manager Cloud Service
- Oracle Utilities Meter Solution Cloud Service
- Oracle Utilities Operational Device Cloud Service
- Oracle Utilities Rate Cloud Service
- Oracle Utilities Work and Asset Cloud Service

This document includes the following:

- Cloud Services Live Operate Services
- Cloud Services Live Operational Guidelines
- Data Fix with Plug-In Driven Batch
- Information Life Cycle Management and Archiving
- Release Management and Regression Testing
- Troubleshooting

Chapter 2

Cloud Services Live Operate Services

This chapter provides information regarding live operate services for Oracle Utilities cloud services, including:

Live Operate Services

Live Operate services are services that Oracle provides to assist you with the ongoing operation of supported Oracle Utilities cloud services, once live and in production. If Live Operate Services are included in your Oracle Utilities cloud service(s), 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 Utilities cloud service(s). 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 Utilities cloud services are performed in accordance with targets.
- Provide access to information about incidents and changes made to any of your Oracle Utilities cloud service environment(s).
- 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 08:00 to 17:00 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 Utilities 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.

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 Utilities 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 Utilities cloud service 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 Utilities cloud service(s).
- 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 Utilities cloud service 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.

Chapter 3

Cloud Services Live Operational Guidelines

This chapter provides guidelines around the operation of Oracle Utilities cloud services once configured to a customer's requirements, including:

- 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

Batch Job Submission

Much of the heavy processing in Oracle Utilities 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 **Background Processes** in the *Administrative User Guide* or the online help for more information.

Batch Job Scheduling

Using Oracle DBMS Scheduler

The Oracle DBMS Scheduler is provided with Oracle Utilities cloud services and is the default job scheduling option.

Refer to **Batch Scheduler Integration** in the *Administrative User Guide* or the 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. Please review the Batch Scheduler Integration for Oracle Utilities Application Framework technical reference paper in Oracle Support (Document ID 2196486.1) 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.

Partial DBMS Updates

Partial updates to the Database Management Schedule (DBMS) chain are not allowed.

Pausing DBMS Jobs

Holding the execution of DBMS jobs is currently not supported.

Scheduling Batch Jobs

Scheduling a batch job to be effective in future is currently not supported.

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 (similar to 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 **Service Health Check** in the *Administrative User Guide* or the online help for more information.

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 Utilities 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.

What is Cloud Service Foundation?

Cloud Service Foundation (CSF) is an add-on companion product for Oracle Utilities cloud services that provides 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). More information about CMA and Process Automation Tool is available in the relevant Administrative User Guide for each cloud service.

Refer to **Content Migration Assistant (CMA)** in the *Administrative User Guide* or the online help and **Process Automation Tool** in the *Oracle Utilities Cloud Service Foundation Administrative User Guide* for more information.

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**.

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 has the ability to 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.

A sample System Health Check Probe like this exists and is available to customers and implementers for educational purpose only. This sample is NOT supported by Oracle Support. See the *Oracle Utilities System Health Check Probe* document on My Oracle Support (Document ID 2711546.1) for more information.

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.

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 Utilities Cloud Services Cloud Operations Guide* for more information about submitting data cloning service requests.

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.

Limits on Time or Size of Certain Services

Oracle Utilities 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 service agreements, which should be done in batch).

Analytics Publisher Limits

The table below lists a number of pre-set limits in Analytics Publisher for reporting:

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

Oracle REST Data Services Limits

The table below lists a number of 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)

Web Service Limits

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

Process	Default	Limit
Web Service Call	1 minute	5 / 15 minutes*

* 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 Cloud 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.

Batch Processing Limits

The table below lists SQL timeouts for batch processing for different types of thread pool workers.

Thread Pool Worker	SQL Timeout
NOCAHCE TPW	15 minutes and 30 seconds*
DEFAULT TPW	15 minutes and 30 seconds*

* The SQL timeout limit for the NOCACHE TPW can be extended upon request *by submitting a Service Request to the Cloud Operations team*. Custom batch processes must be tested with production data to ensure they adhere to the SQL timeouts listed above. For example, the SQL timeout for a custom batch process run using the DEFAULT TPW must not exceed 15 minutes and 30 seconds. Refer to Reviewing and Tuning SQL Statements for guidelines.

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 are able to 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 **Debug Mode** in the *Administrative User 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 **Debug Mode** in the *Administrative User 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).

Chapter 4

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 chapter provides guidelines for an approach to these types of fixes using Plug-In Drive batch processes, including:

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

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 **Plug-in Driven Background Processes** in the *Administrative User 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 Utilities Cloud Services Cloud Operations Guide* for more information.

Plug-In Driven Batch Components

For those unfamiliar, a plug-in driven batch is a batch control defined with a 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)

Optionally you can either use SQL Developer WEB or BI publisher define a zone for testing the query if you are working in a cloud environment.

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.

Use Case

A large number of devices in Customer Cloud Service or Meter Solution 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-SMART-MTR 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).

Step One: Create the Batch Control

Creating the batch control begins by duplicating the OUAF delivered batch control F1-PDBG - 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

Step Two: Create a Select Records Algorithm

This algorithm should be created by duplicating the sample OUAF algorithm type F1-PDB-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: 1. :F1 is the device type code that will be provided in the batch parameters 2. :F2 is the head end that will be provided in the batch parameters 3. :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. 4. :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.

Parameter	Sequence	Value	Comments
Batch Category	2	THDS	To take advantage of multi-threading 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.
Key Field	3	D1_DEVICE_ID	This is identifying which of the returned fields by the query is being used in the threading strategy.

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()
```

```
            Usage_DTO usageDTO = usage.getDTO()
            //  update Bill Segment on Usage via DTO
            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(UnableToCompleteBillActionLookup.co
nstants.SHOW_ERROR)

         bill.complete(billCompletionInputData)

      }
```

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.

Chapter 5

Information Life Cycle Management and Archiving

This chapter describes the use of Information Lifecycle Management (ILM) and archiving with Oracle Utilities Cloud Services, including:

- Information Lifecycle Management
- Customer Responsibilities - Post Go Live

Information Lifecycle Management

Information Lifecycle Management is a set of techniques and technologies available from Oracle that assist in managing the lifecycle of data to support business needs and minimize storage costs. The key to Information Lifecycle Management is working with the business to ensure that data that the business regards as active is available on appropriate hardware whilst data that is less active or dormant, in terms of business activity, is managed in an effective way in terms of storage costs.

Data in the product is added and updated by the business on an ongoing basis. Over time, this data grows and the business activity on individual data will vary with its timeliness, business activity and data retention policies. As data becomes less active in terms of the business, it needs to be stored in a cost-effective way to ensure cost minimization whilst allowing the business to access the data as needed for analysis or auditing purposes.

As the Oracle Database stores and manages data on an ongoing basis, therefore it is logical that Information Lifecycle Management uses the inbuilt facilities and various database options to offer effective storage management solutions to manage that data.

Once the customer is live, there are several tasks to be performed by the customer. These tasks include creation of future partition, apply compression on existing partitions, running monitor process to check archive data eligibility based on defined retention period, copy the archive data to the cloud object storage and when ready, drop the partition.

Oracle also provides monthly database storage reports which includes database storage size, high volume tables and some partitions information that will enable the customer to make the ILM-related decisions.

Customers will need to set up a batch stream for the 'Add Partition' job (F1-ILMAD) and the add sub-partition job (F1-ILMSV) on a monthly basis to ensure that new monthly partitions are created.

Customers will need to set up a batch stream for partition compression jobs for measurement tables (F1-ILMMC) and non-measurement tables (F1-ILMNC).

At this time, customers should simply ensure partitions are kept up to date for eventual archive.

Oracle recommends that customers should perform ILM testing in their Test environment before running ILM batch jobs in their Production environment.

Exclusions: WACS does not yet support ILM (i.e. W1 prefix tables are not partitioned).

Customer Responsibilities - Post Go Live

Oracle recommends that customers configure batch job streams to run the following batch jobs on a regaular basis to support ILM:

- **ILM Automation - Add Partition** (F1-ILMAD): This process creates partitions for ILM-enabled tables for the next month. The process splits the maximum partition into the next month (current month +1) and copies the table stats from the previous month to the new partition and the maximum partition. This is critical to ensuring performance when the next month begins. This must be run once per month but can be run more frequently without causing harm.

  F1-ILMAD Sample Parameters:

Parameter Name	Description	Parameter Value	Detailed Description	Required
tableOwner	Table Owner	CISADM	Schema owner name of the tables to be processed.	☐
startDate	Start Date (YYYY-MM-DD)	2022-01-01	When no value is supplied, the batch will use process date. Based on the start and end date values, the batch will either create partitions for the next month or for months between the start and end date (inclusive).	☐
endDate	EndDate (YYYY-MM-DD)	2022-01-31	When no value is supplied, the batch will use process date. Based on the start and end date values, the batch will either create partitions for the next month or for months between the start and end date (inclusive).	☐
dataFileLocation	Data File Location		When populated, value overrides the default directory where the tablespace data file will be created.	☐
tableNames	Table Names (comma delimited)		List of table names to be processed. When no value is defined, all ILM enabled and date partitioned non-ILM tables will be processed	☐
DIST-THD-POOL	Thread Pool Name	NOCACHE	Thread pool name to use if the DEFAULT thread pool is not desired.	☐
maxErrors	Override maximum errors		Enter a value here to override the maximum number of errors allowed before the run is terminated.	☐

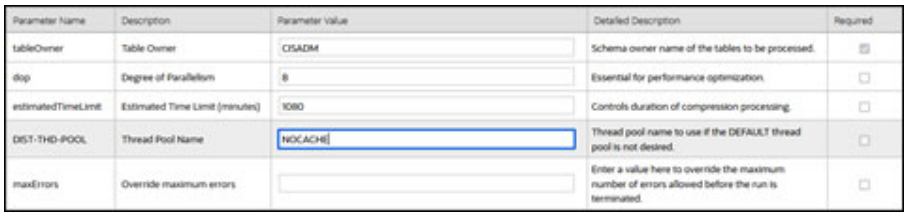
  Note: This batch job should be run with DIST-THD-POOL batch parameter with value of NOCACHE.

- **ILM Automation - New Subretention Value** (F1-ILMSV): This process creates subpartitions for Initial Measurements, Activities, and Device Events, based on specific sub-retention values defined in the ILM Configuration - MDM Sub-retention master configuration. Note that Sub-retention is optional. If used, this should be run at the same time as F1-ILMAD.

- **ILM Automation - Measurement Compression** (F1-ILMMC): This process changes the compression level from Advanced to Hybrid Columnar for Measurement data that is at least two months old which results in significant disk space saving. This compression job reorganizes the data and improves query performance. Parameters used by this process include: Table Owner, Degree of

Parallelism, and Estimated Time Limit. This batch control should be part of monthly batch job streams so that its run once per month and during off peak / non-critical hours.

F1-ILMMC Sample Parameters:

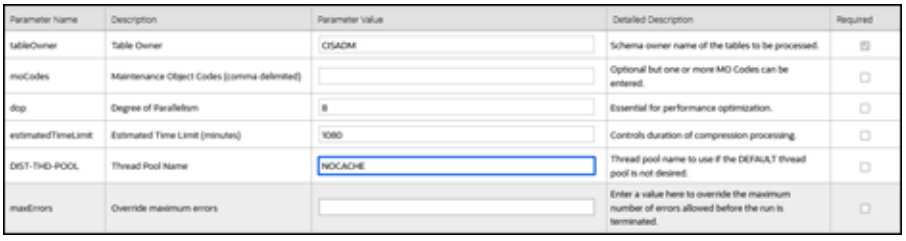


Parameter Name	Description	Parameter Value	Detailed Description	Required
tableOwner	Table Owner	CISADM	Schema owner name of the tables to be processed.	☑
dop	Degree of Parallelism	8	Essential for performance optimization.	☐
estimatedTimeLimit	Estimated Time Limit (minutes)	1080	Controls duration of compression processing.	☐
DIST-THD-POOL	Thread Pool Name	NOCACHE	Thread pool name to use if the DEFAULT thread pool is not desired.	☐
maxErrors	Override maximum errors		Enter a value here to override the maximum number of errors allowed before the run is terminated.	☐

Note: This batch job should be run with DIST-THD-POOL batch parameter with value of NOCACHE.

- **ILM Automation - Non-Measurement Compression** (F1-ILMNC): This process changes the compression level from Advanced to Hybrid Columnar for non-Measurement partitions at least two months old and results in significant disk space savings. Parameters used by this process include: Table Owner, Maintenance Object Codes (used to define specific maintenance objects for compression), Degree of Parallelism, and Estimated Time Limit. This batch control should be part of monthly batch job streams so that its run once per month and during off peak / non-critical hours.

  F1-ILMNC Sample Parameters:



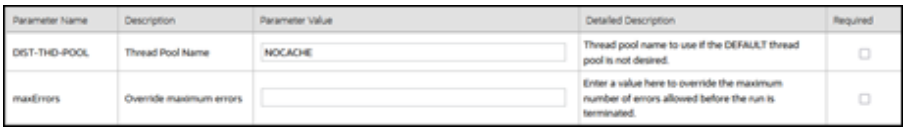
Parameter Name	Description	Parameter Value	Detailed Description	Required
tableOwner	Table Owner	CISADM	Schema owner name of the tables to be processed.	☑
moCodes	Maintenance Object Codes (comma delimited)		Optional but one or more MO Codes can be entered.	☐
dop	Degree of Parallelism	8	Essential for performance optimization.	☐
estimatedTimeLimit	Estimated Time Limit (minutes)	1080	Controls duration of compression processing.	☐
DIST-THD-POOL	Thread Pool Name	NOCACHE	Thread pool name to use if the DEFAULT thread pool is not desired.	☐
maxErrors	Override maximum errors		Enter a value here to override the maximum number of errors allowed before the run is terminated.	☐

Note: This batch job should be run with DIST-THD-POOL batch parameter with value of NOCACHE.

- **ILM Crawler Initiator** (F1-ILMIN): The ILM Crawler Initiator process initiates the individual ILM Crawler batch controls as defined by the ILM Crawler Batch Control and ILM Retention Period in Days options on each maintenance object. his batch control should be part of monthly batch job streams so that its run once per month and during off peak / non-critical hours.

  Individual ILM crawler batch controls invoke eligibility algorithms to determine if records are ready to be archived. Many maintenance objects use the ILM Eligibility Based on Status (F1-ILMELIG) algorithm to determine eligibility.

  F1-ILMIN Sample Parameters:



Parameter Name	Description	Parameter Value	Detailed Description	Required
DIST-THD-POOL	Thread Pool Name	NOCACHE	Thread pool name to use if the DEFAULT thread pool is not desired.	☐
maxErrors	Override maximum errors		Enter a value here to override the maximum number of errors allowed before the run is terminated.	☐

Note: This batch job should be run with DIST-THD-POOL batch parameter with value of NOCACHE.

- To copy the archived data to Oracle Cloud Object Storage and to drop the partition, customer is required to raise a Service Request with Oracle Support.

Refer to **Information Lifecycle Management** in the *Administrative User Guide* for more details regarding ILM.

Chapter 6

Release Management and Regression Testing

This chapter describes release management processes and regression testing of Oracle Utilities Cloud Services, including:

- Cloud Service Release Management
- Regression Testing through Utilities Testing Accelerator
- Regression Testing Recommendations

Cloud Service Release Management

Oracle Utilities cloud services include software releases & updates, Infrastructure Updates, Maintenance Packs and Hotfixes which Customer are obligated to take in timely manner.

Software Releases & Updates

Staying current with software updates also directly benefits the customer:

- Access to latest functionality, means needing fewer customizations.
- New features are typically 'opt-in' so you can choose when to configure & use them
- Having all available patches applied, means discovering fewer problems yourselves, and makes it easier for customer support to replicate any issues you do experience
- Having all security patches applied, means less risk of security breaches.

Oracle Utilities plans for three releases a year: April, August, December

- Releases introduce new features and may make improvements and changes to existing functionality
- The releases are labeled using the format YY followed by A, B or C (A=April, B=August, C=December). For example, the August 2021 release is called '21B'.
- Product documentation is made available in advance of each update, including a New Feature Summary about one month before general availability.

Each release has a 'lifespan' of one year – i.e. end of life is one year after release,

Customers must operate a Generally Available release of the Oracle Cloud Service. General Availability and End of Life (EOL) dates are published in the Oracle Utilities Program Documentation.

A new release is applied as an update of each environment (non-production first for testing), retaining all configuration and data.

Infrastructure Updates

The underlying technology platform receives monthly updates

- This technology platform is independent of the actual software release (e.g. 21B) that a customer is running
  - This includes technologies like database, operating system, etc.
  - This also includes services like monitoring, alerting, logging, etc.
- All customers receive these platform updates every month
- No exceptions

Maintenance Packs

For each release, Oracle Utilities provides bug fixes and patches.

- The standard mechanism for this is a Maintenance Pack (MP), which bundles up product fixes on the latest release on a monthly basis.
  - Uptake of Maintenance Packs is required by all customers still implementing the service (in other words, more than 4 months from go-live - we call this the 'implementation' or 'fast' lane)
  - Each Maintenance Pack is identified with a sequential number (MP 1 through MP 8)
  - Release notes for each Maintenance Pack are published in My Oracle Support with detailed information about each fix – what's changed, how to verify it.
  - Maintenance Packs are pushed out to chosen environments on the 1st weekend (for early adopter environments) and 3rd weekend (for later adopter environments including 'Production') of each month.

Hot Fixes for Critical Issues

For more time-sensitive critical application fixes there is a 'hot fix' mechanism that can provide product patches off the monthly maintenance cycle.

- Once Maintenance Packs stop for a release, we move to a 'hot fix-only' model (known as the 'cutover' lane and 'production' lane)
- Note that hot fixes are cumulative to the latest Maintenance Pack, so customers are not able to 'pick and choose'
  - If a customer needs a fix they will get all hot fixes available up to that point

    - If a customer does not need any patches after MPs end, they do not have to take any
- Hot fixes are not used to backport changes; they are intended to address only critical issues which customers have reported.
- Hot fixes are sometimes issued while Maintenance Packs are still coming out (and are then rolled into the next Maintenance Pack)
- Hot fixes will continue as necessary to the End of Life of the release (12 months from General Availability).

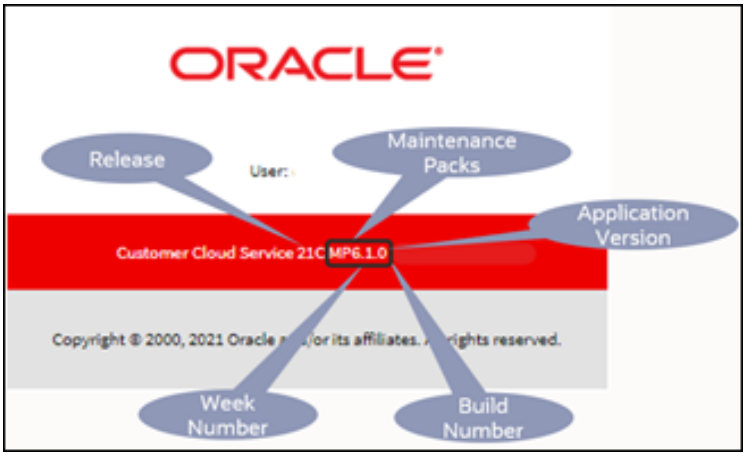
Oracle Recommended Approach to Planning for Updates:

- Oracle recommends a 'three lane' approach to support customers thru the different stages of their lifecycle:
- **Implementation** or **Fast** lane: During most of the implementation project – early adoption of new releases as they become available
    - Uptake and use latest release with latest functionality, full maintenance packs each month
- **Cutover** lane: When approaching Go-live – use the extended 'runway' on the target release to stabilize
    - Slow down rate of change and stabilize with any necessary hotfixes
- **Production** lane: For live production customers only
    - Uptake releases late, take only critical hot fixes, stay on 4-month update cycle

For more details, refer to the *Oracle Utilities Cloud Services Upgrade Policy Guide* on the Oracle Utilities Documentation page.

Application Version Displayed in About Window:

The application "About" window displays information about the current application version. The image below shows an example of the "About" window.



1. The "About" window displays data as outlined below (this example is for Customer Cloud Service 21C MP6.1.0):

Application Version				
Service Name	**Release**	**Maintenance Pack**	**Week Number**	**Build Number** (0 for new build, 1 for additional offcycle hotfix or re-packaging)
Customer Cloud Service	21C	MP6	1	0

2. The Application version will comprise the Maintenance Pack, Week number, and build. Week numbers will start from the day MP is released.
3. The application version includes week number following the maintenance pack number. (For example, a hot fix released for MP6 in Week-1 would show as MP6.1.0, Week-2 would be MP6.2.0).
4. Build numbers are the incremental numbers reflecting additional hot fixes or re-packaging (if needed) followed by the week number. (For example, an additional hot fix released for MP6 in Week-1 would show as MP6.1.1).
5. In case where NO hot fix released in a given week, the week number would be skipped. For example, if we release hot fixes for week-3 but skip week-2, the About window will display MP6.3.0 and there won't be a MP6.2.0.
6. For last maintenance pack (MP8), the hot fix number will continue to increment after week-4. For example, MP8.5.0, MP8.6.0 and so on.

Regression Testing through Utilities Testing Accelerator

Oracle Utilities Testing Accelerator (UTA) comprises test automation accelerators for the automated testing of Oracle Utilities applications. It is a framework based on Java and Selenium for creating the web services and user interface automation scripts.

Oracle Utilities Testing Accelerator contains out-of-the-box product-specific components used to build new test flows in Oracle Utilities Testing Accelerator Workbench to test the Oracle Utilities applications. These out-of-the-box components correspond to specific business entities, such as business objects, service scripts, or business services used for interfacing with the application. Users can use these components as available or can extend them. Users can also create new components to be used to create flows.

Refer to the Oracle Utilities Testing Accelerator documentation for more information.

Regression Testing Recommendations

As part of implementation, customers should use Utilities Testing Accelerator (UTA) in several testing phases and after go-live must have several established UTA process flows created to verify the business process in an automated way. Oracle recommends continuing using the UTA for any upgrade performed in their environment.

Upon receipt of any hot fix, new maintenance pack, or new release, UTA is the best way to perform regression testing on an upgraded environment. Aside from using UTA for regression testing, Oracle also recommends the following:

- Customer taking new maintenance packs and or hot fixes are required to review the Product Fix Documents (PFDs) available on My Oracle Support. These documents contain changes included in the hot fix along with some testing scenario that you can create in the UTA as process flows.
- Customers taking new releases are required to perform the following steps:
    - Review the Oracle Cloud Application Update Readiness portal.
        - Review the What's New (under Essential Content - an Excel sheet is also available for further details)
        - Review the New Feature Summary (under Supplemental Content)
        - Review the new feature details, as well as its impact and where it will be used.
        - Review the **Database Changes** feature to be aware about new database objects and changes made to existing objects
        - Review the **Important Actions and Considerations** sections for removed and replacement features with time frame.
    - Access the **Extensions Dashboard** portal in the cloud application to review your custom extensions to ensure that your testing includes your customizations. Refer to **Extensions Dashboard** in the *Administrative User Guide* for more information about this portal.
    - Review the new release videos demonstrating the new feature and how to configure your application to use it. Access these videos via the **Videos** link in the left navigation pane on Oracle Help Center.
    - Review the Utilities Testing Accelerator (UTA) documentation. If a UTA upgrade is needed, perform the UTA upgrade first.
    - Run the F1-CAGVY batch process in your application to recompile any custom groovy code. Make sure to review the batch run tree and resolve errors in custom code, if found.
    - Review user security by comparing the existing application services, and add any appropriate new application services to enable new features.
    - Run the UTA process flows to verify that existing business processes are running as configured. That includes verification that new bills are being created correctly (for Customer Cloud Service, Billing Cloud Service, and Customer Care and Billing Cloud Service).
    - Configure new features (as required) and perform initial testing.
    - Create new UTA process flows for new features and/or processes.

- Perform the regression testing multiple times in both isolation and with integrations enabled.
- Make sure the batch job streams are executed in the same manner as in production. Review batch runs for any errors.

Chapter 7

Troubleshooting

This chapter provides troubleshooting guidelines for Oracle Utilities cloud services, including:

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

Database Monitoring Using SQL Developer

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 Utilities 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.

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

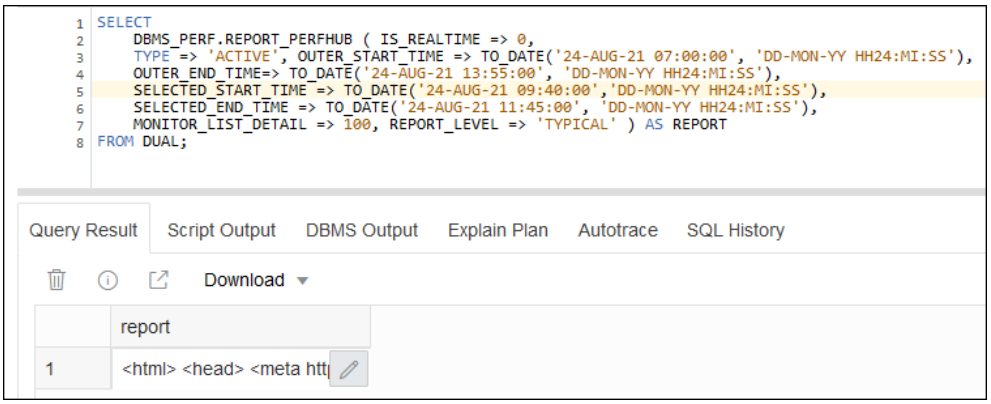
Use the following procedure to generate the Performance Hub Report:

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('24-AUG-21
07:00:00', 'DD-MON-YY HH24:MI:SS'),
    OUTER_END_TIME=> TO_DATE('24-AUG-21 13:55:00', 'DD-MON-YY
HH24:MI:SS'),
    SELECTED_START_TIME => TO_DATE('24-AUG-21 09:40:00','DD-MON-YY
HH24:MI:SS'),
    SELECTED_END_TIME => TO_DATE('24-AUG-21 11:45:00', '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:



2. Export the output in xml format (select **Download** in the **Query Result** section) 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

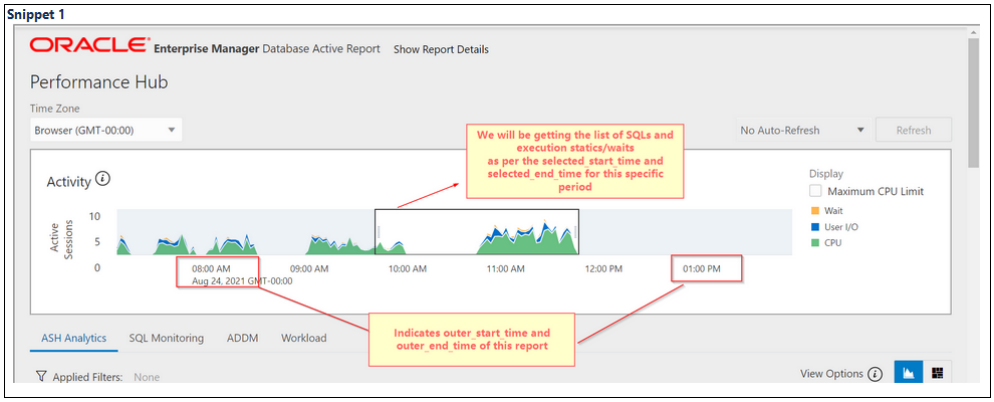
The following table lists parameters that can be used when running the Performance Hub Report:

Paremeter	Description
is_realtime	If 1, then real-time. If NULL (default) or 0, then historical mode. • When real-time data is selected (1), more granular data is presented (because data points are available every minute). • When historical data is selected (0), more detailed data (broken down by different metrics) is presented, but the data points are averaged to specified intervals (typically an hour).
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.
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)

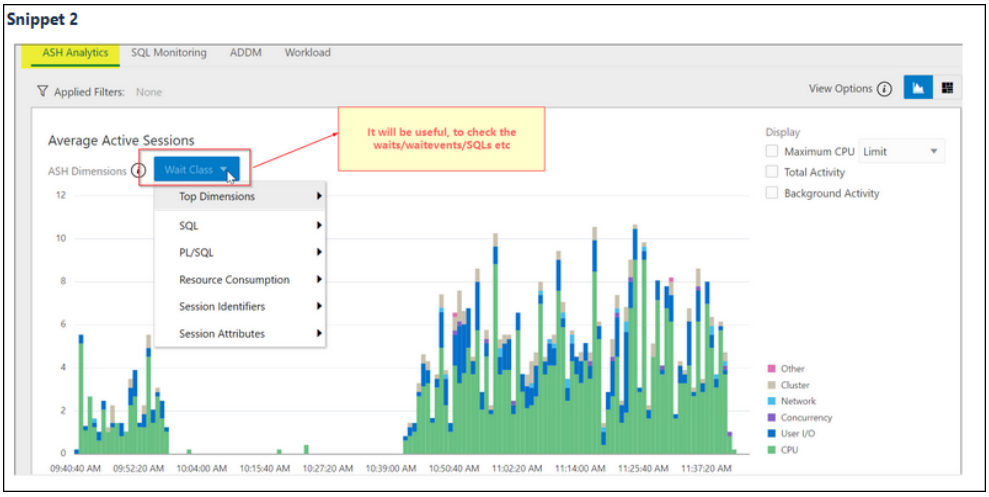
Paremeter	Description
selected_start_time	Start time period of selection. If NULL (default) • If is_realtime=0, then 1 hour before selected_end_time • If is_realtime=1, then 5 minutes before selected_end_time
selected_end_time	End time period of selection. If NULL (default) • If is_realtime=0, then latest AWR snapshot • If is_realtime=1, then current time
inst_id	Instance ID to 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
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.

Performance Hub Report Sample Output

Snippet 1



Snippet 2



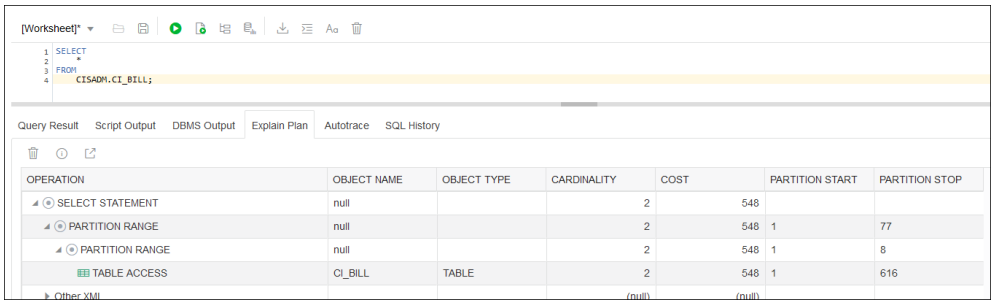
Important Notes

- 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).

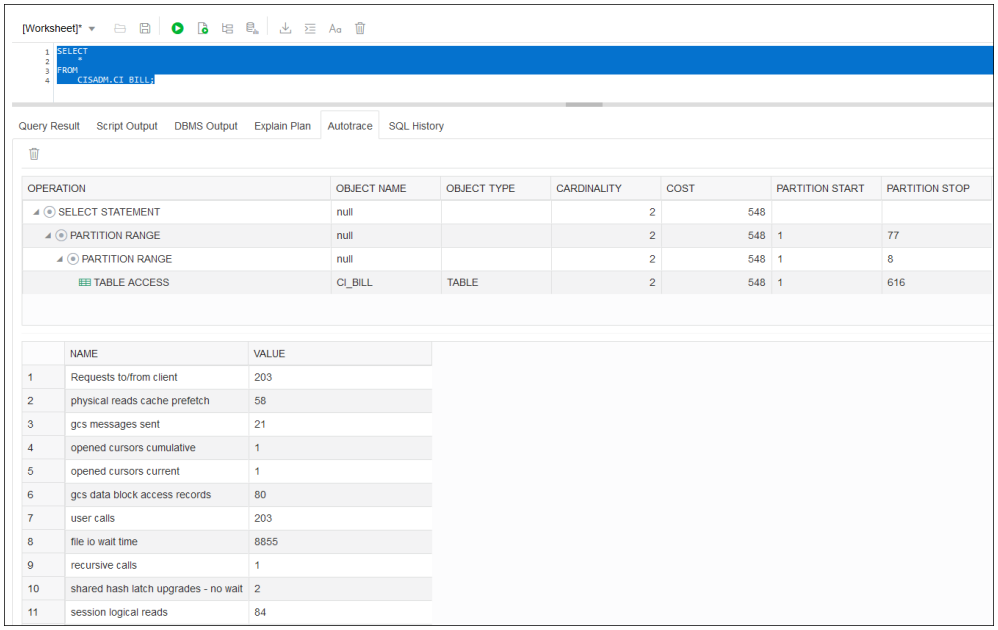
Reviewing and Tuning SQL Statements

SQL statements retrieved using the Performance Hub Report can be reviewed and tuned using the SQL Developer web (ORDS) Explain Plan and AutoTrace functionality. The following screen shots illustration these functions.

Explain Plan:


[Worksheet]*
SELECT
*
FROM
CISADM.CI_BILL;
Query Result
Script Output
DBMS Output
Explain Plan
Autotrace
SQL History
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
Other XML
(null)
(null)

AutoTrace:


[Worksheet]*
SELECT
*
FROM
CISADM.CI_BILL;
Query Result
Script Output
DBMS Output
Explain Plan
Autotrace
SQL History
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
203
8
file io wait time
8855
9
recursive calls
1
10
shared hash latch upgrades - no wait
2
11
session logical reads
84

Running and Troubleshooting Batch Processing

This section provides guidelines for running and troubleshooting batch processing with Oracle Utilities cloud services, including:

Introduction

Batch jobs are used for several key purposes in Oracle Utilities 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.

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 or F1-ILMAD 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.

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) and Information Lifecycle Management (ILM) related batch jobs such as K1-RIUSP, F1-ILMAD, 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.

Batch Performance Factors

Batch performance is dependent on a number of 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.

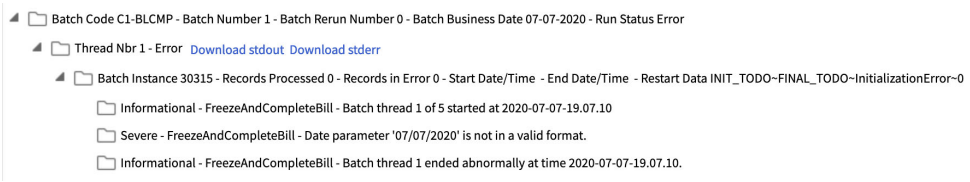
Frequently Asked Questions for Common Batch Issues

Q1) 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, Instance(s) 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:



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:

```
2021-05-14 13:52:40.013-0700 [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
```

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:

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

```
2021-08-04 11:39:10.267-0700 [83] 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 ToDo 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.

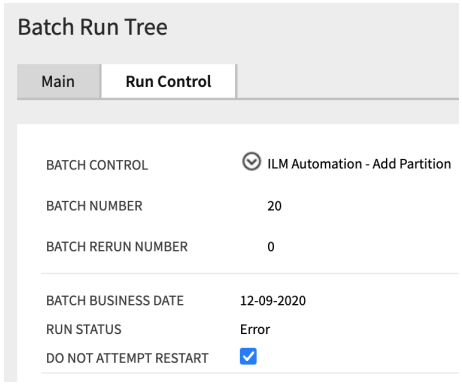
Important TIPS:
If the logs are not pointing towards the exact error, logs are emptied or logs don't guide , please follow the steps below:

- 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

- 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 **Prepare Issue Details** in the *Administrative User Guide*), batch job submission with parameters, batch run tree (expanded), and both logs (STDERR and STDOUT).

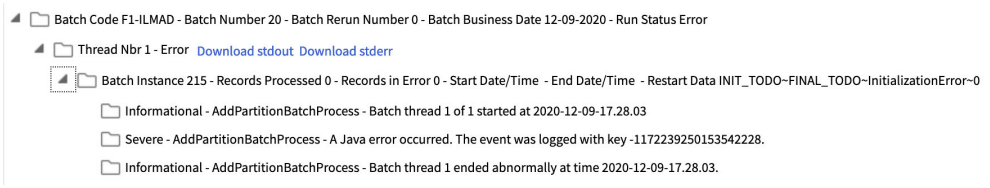
  Note: If the errored batch is custom-built, please contact the project implementation / support team.

Q2) 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 re-start. 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).



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.

Q3) 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.



Above is a sample Batch Run Tree with a Java error, please follow the 'Important TIPS' for troubleshooting shown under Q1.

Q4) 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. In order to restart, please run the application in debug mode where TPW restart option is available (as seen below)

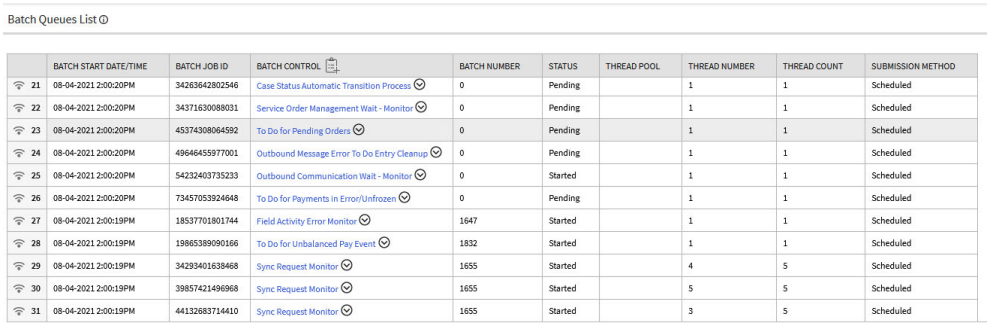
- Check the desired execution date/time, it should not be in future.

Q5) 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.

Q6) I want to submit a batch, but I don't know what batches are currently running?

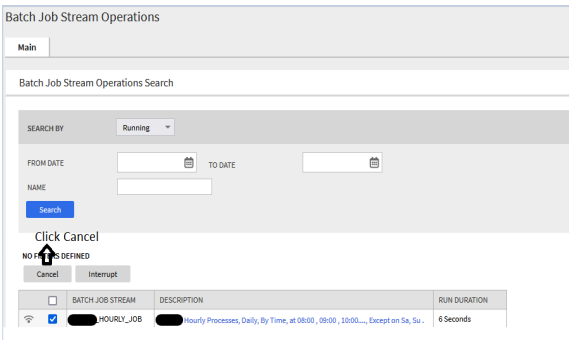
To check this, go to the Admin → B → Batch Queue Portal page and verify if any running jobs. If you see batch jobs already showing in pending status that means you have consumed all the available threads allocated and no capacity available to submit any more batches.



	BATCH START DATE/TIME	BATCH JOB ID	BATCH CONTROL	BATCH NUMBER	STATUS	THREAD POOL	THREAD NUMBER	THREAD COUNT	SUBMISSION METHOD
21	08-04-2021 2:00:20PM	34263642802546	Case Status Automatic Transition Process	0	Pending		1	1	Scheduled
22	08-04-2021 2:00:20PM	34371630088031	Service Order Management Wait - Monitor	0	Pending		1	1	Scheduled
23	08-04-2021 2:00:20PM	45374308064592	To Do for Pending Orders	0	Pending		1	1	Scheduled
24	08-04-2021 2:00:20PM	49646455977001	Outbound Message Error To Do Entry Cleanup	0	Pending		1	1	Scheduled
25	08-04-2021 2:00:20PM	54232403735233	Outbound Communication Wait - Monitor	0	Started		1	1	Scheduled
26	08-04-2021 2:00:20PM	73457053924648	To Do for Payments in Error/Unfrozen	0	Pending		1	1	Scheduled
27	08-04-2021 2:00:19PM	18537701801744	Field Activity Error Monitor	1647	Started		1	1	Scheduled
28	08-04-2021 2:00:19PM	19865389090166	To Do for Unbalanced Pay Event	1832	Started		1	1	Scheduled
29	08-04-2021 2:00:19PM	34293401638468	Sync Request Monitor	1655	Started		4	5	Scheduled
30	08-04-2021 2:00:19PM	39857421496968	Sync Request Monitor	1655	Started		5	5	Scheduled
31	08-04-2021 2:00:19PM	44132683714410	Sync Request Monitor	1655	Started		3	5	Scheduled

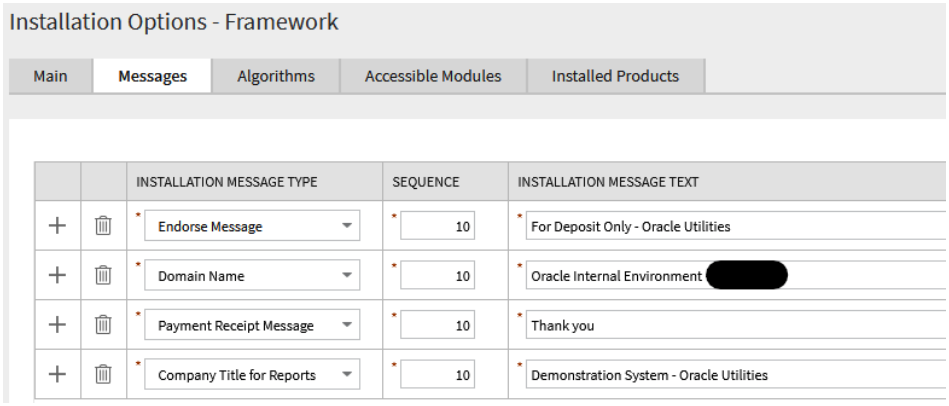
Q7) I am trying to cancel the batch submitted by DBMS Scheduler, it is giving me error. How to cancel jobs submitted by DBMS Scheduler?

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 can not be cancelled through the cancel option. In order to cancel such batch(s), you have to 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.

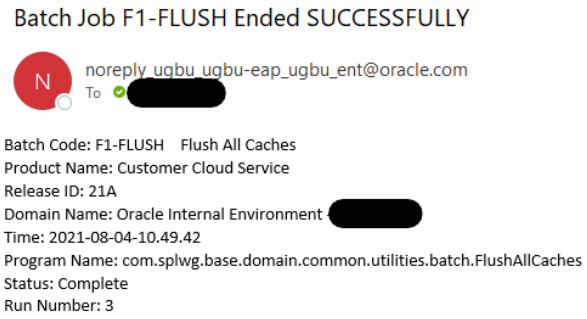


Q8) Why do I see multiple instances of F1-BTMON batch in batch job submission?
F1-BTMON is the batch probe - which is run automatically as part of the cloud service monitoring of each environment, to verify that batch is able to execute. This job does a small amount of transitory database access, and 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.

Q9) 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.



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

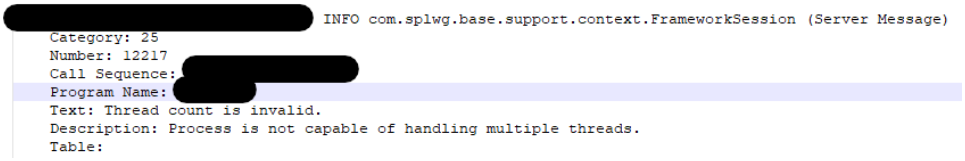


Above is a sample batch notification email showing the Domain Name ("Oracle Internal Environment").

Q10) 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:



Q11) 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.

Q12) 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?

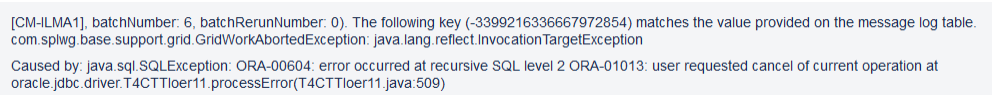
We have seen a couple reasons for this situation - sometimes it really is not 'stuck' - but it is doing a large amount of work on a single record (for instance billing an Account with many Service Agreements). If the record count on the thread remains at the same number for more than 5 minutes, you may 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.

Q13) The logs for my batch job are empty - what should I do?

There are a handful of jobs that will not produce logs - these are mostly related to ILM processing (F1-ILMAD, which others) that call stored procedures. But in general 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.

Q14) 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 and or ILM related) 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.



The above sample log entry typically indicates a timeout issue.

Q15) 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.

Q16) 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, in order to keep the output manageable.

Q17) 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 meta-data. 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 actually 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 particular 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 have to do the 'flush' to empty and rebuild certain caches so that the new/changed values are accessible to everyone.