Oracle Utilities Cloud Services

Live Operations Guide

Release 26.4
G50324-03
April 2026

ORACLE

Oracle Utilities Cloud Services Live Operations Guide, Release 26.4

G50324-03

Contents

Index

1

Introduction

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 Customer Program Management Cloud Service
- Oracle Utilities Market Settlements Management Cloud Service
- Oracle Utilities Meter Solution 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
- Oracle Utilities Break Glass
- Data Fix with Plug-In Driven Batch
- Information Lifecycle Management
- Release Management and Regression Testing
- Disaster Recovery
- GoldenGate Replication
- Troubleshooting

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:

- Provide you with access to a Customer Success Manager ("CSM") who will deliver customer success services. To learn more about what you can expect from a CSM, please visit https://www.oracle.com/customer-hub/utilities/working-with-customer-success/.
- 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 suitably qualified support personnel who are able to extract, interpret, analyze and respond to information provided in the Performance Hub reports.
- 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, including, but not limited to, business/access logs and Performance Hub reports.
  - 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.

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

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 SaaS Batch 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

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

The SaaS Batch Scheduler is a feature in Oracle Utilities Cloud Service Foundation that provides a SaaS batch scheduler engine and user interfaces. It facilitates customers and implementers to define, manage and schedule stream of batch jobs to run at periodic intervals using a user friendly interface. The SaaS batch scheduler engine is responsible for scheduling and processing the stream of batch jobs. It has a processing loop that runs every X seconds (configured by default at 10 seconds) to perform these operations. In addition, users can

monitor the progress of the streams, each of the steps and also perform required exception handling. The SaaS batch scheduler engine processes the stream of batch jobs per scheduled time, within that minute.

Pausing Scheduler Batch Streams

Holding the execution of Scheduler Batch Streams is fully supported.

Future Scheduler Batch Streams

Scheduling a Scheduler Batch Streams to be effective in future is fully 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 Scheduler Batch 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 – Scheduler Batch Stream:

- K1SBSNSAE: Scheduler Batch Stream Not Started As Expected
- K1SBSJRTL: Scheduler Batch Stream Job Ran Too Long (Relative Run)
- K1-SBSD-LVSV: Scheduler Batch Stream Has Failed

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

Improved Batch Processing Through Fine-Grained Thread Submission

Oracle Utilities Cloud Services include mature and robust batch processing features to achieve the following goals:

- Simplify the configuration of batch operations by enabling jobs of various types to safely share resources when submitted concurrently.
- Shorten job completion time by preventing outlier late-completing threads of work.
- Improve utilization of entitled threads of capacity to increase environments' overall throughput of business workloads.

This section provides an overview of batch architecture included in Oracle Utilities Cloud Services. It is intended solely to help you assess the business benefits of the batch processing architecture. This includes the following:

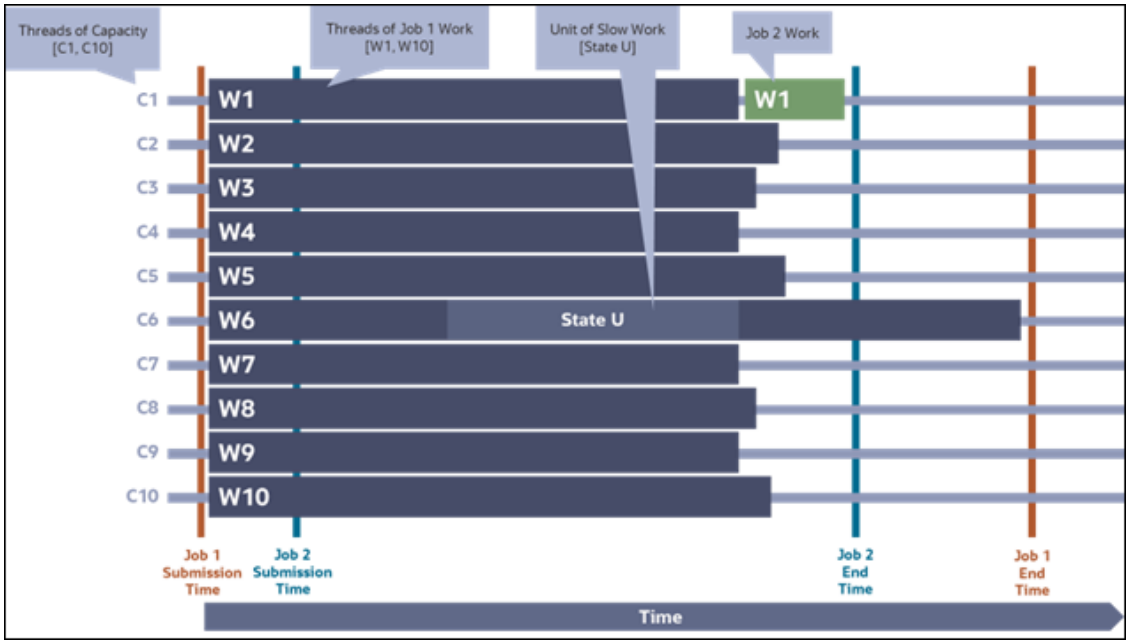
Background: Threads of Work vs. Threads of Capacity

It has long been the case when batch jobs are submitted that a thread count parameter is specified to tell the batch runtime the number of chunks that the job's work should be split into for parallel execution. We refer to these chunks "threads of work". That is, the thread count describes the number of threads of work for a job. Each thread of work is then assigned by the system to run on a single thread of system capacity whenever sufficient capacity exists. We refer to these units of system capacity "threads of capacity". In summary, the thread count on a job specifies the number of threads of work that the job will be split into and then based on the current usage and capacity of the environment those threads of work are matched to threads of capacity at which point work is performed. Those same concepts have been extended to substantially improve the efficiency and overall experience of running batch jobs in Software-as-a-Service (SaaS). The remainder of this section provides details.

Coarse-Grained Thread Submission and Related Difficulties

It is common practice today for jobs to be submitted with the expectation that immediately after submission each thread of work is matched to a free thread of capacity, the threads run for some time and then after the last thread completes then the job is completed. Consider the diagram below which illustrates the coarse-grained submission of two jobs ("Job 1" and "Job 2") followed by descriptions of common difficulties associated with this type of submission.

Figure 3-1 Coarse-Grained Submission - Two Jobs



Difficulty #1: Delayed Job Completion from Thread Throughput Imbalances

In this example, Job 1 is submitted with a thread count of 10. Notably, thread #6 of Job 1 contains a unit of work labeled "State U" representing some unit of work requiring far more time to complete than other units. The name is chosen in reference to accounts often related to universities which can be particularly complex. Correspondingly, this complex unit of work causes thread #6 to finish at a time much later than the other threads and hence the completion time of the overall job is delayed while waiting for that single thread. Any jobs depending on the completion of Job 1 are likewise delayed.

Similar difficulties although usually of a lower typical impact arise because a particular thread of capacity runs on a computer suffering greater resource contention (such as network, CPU) than other computers and therefore the thread of work finishes later than others which causes the overall job to finish later.

The root of the difficulty is that each thread of work has committed itself to run on a single thread of capacity throughout the duration of the job. Hence no rebalancing of the workload is possible once the job starts.

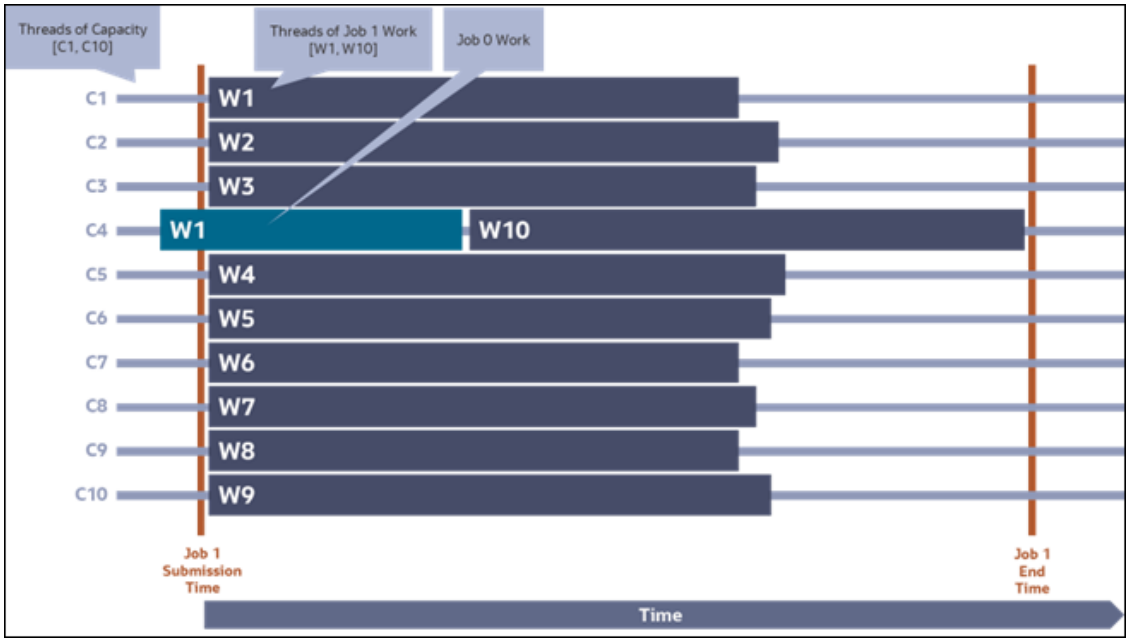
Difficulty #2: Resource Starvation

Consider Job 2 in the diagram which was submitted a short time after Job 1 was submitted. Despite Job 2 having only one thread of work and running comparatively quickly, Job 2 is blocked from running for a long time. Only when a thread of Job 1 completes can Job 2 run at all. This is an example of the well-known challenge in task processing systems, the problem of "resource starvation."

Difficulty #3: Delayed Job Completion from Thread Capacity Shortfall

Consider Figure 3-2 in which an already running job (Job 0) is utilizing one thread of capacity when the parallel job (Job 1) is submitted. Because the single thread (W1) of Job 0 makes one thread of capacity unavailable the result is that the overall completion of job 1 is delayed for the full duration of this capacity shortfall. Coarse-grained job submissions are often incapable of handling thread capacity shortfalls without delays to job completion time.

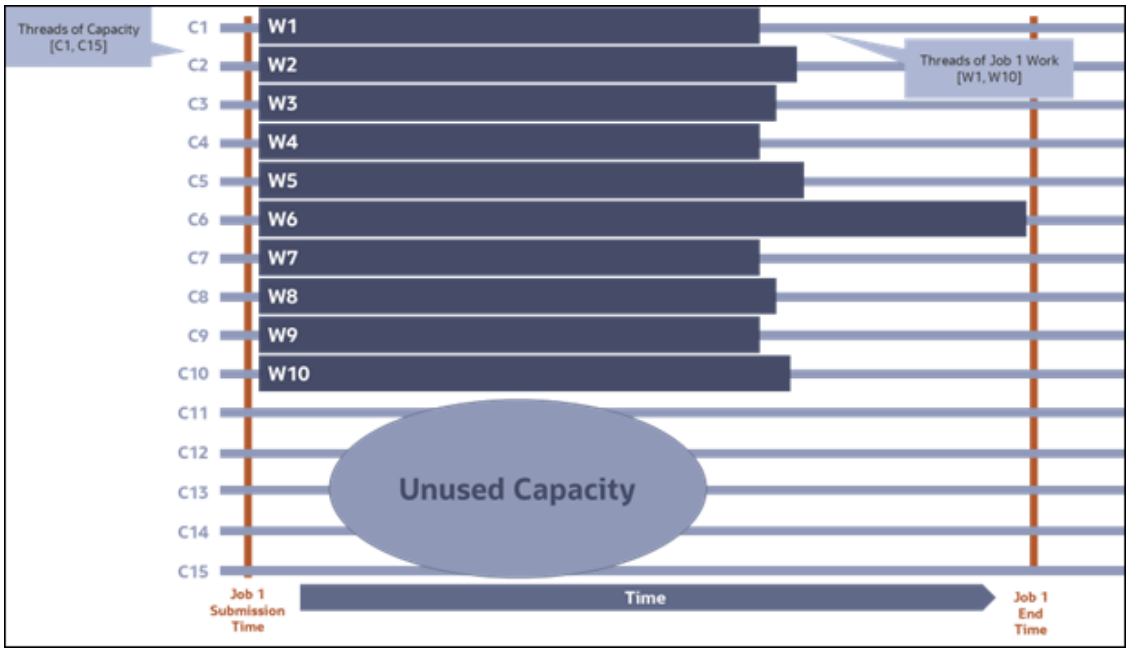
Figure 3-2 Coarse-Grained Submission - Delayed Job Completion



Difficulty #4: Unused Capacity

To avoid difficulties #2 and #3 and to make possible the scheduling of other types of jobs along with parallel jobs like Job 1 it is common to lower the thread count of jobs like Job 1 such that spare capacity is maintained. However, as Figure 3-3 below demonstrates this leads to unused capacity which might otherwise be used to lower the completion time of Job 1.

Figure 3-3 Coarse-Grained Submission - Unused Capacity



The next section describes how difficulties #1 - #4 are solved through Fine- Grained Thread Submission.

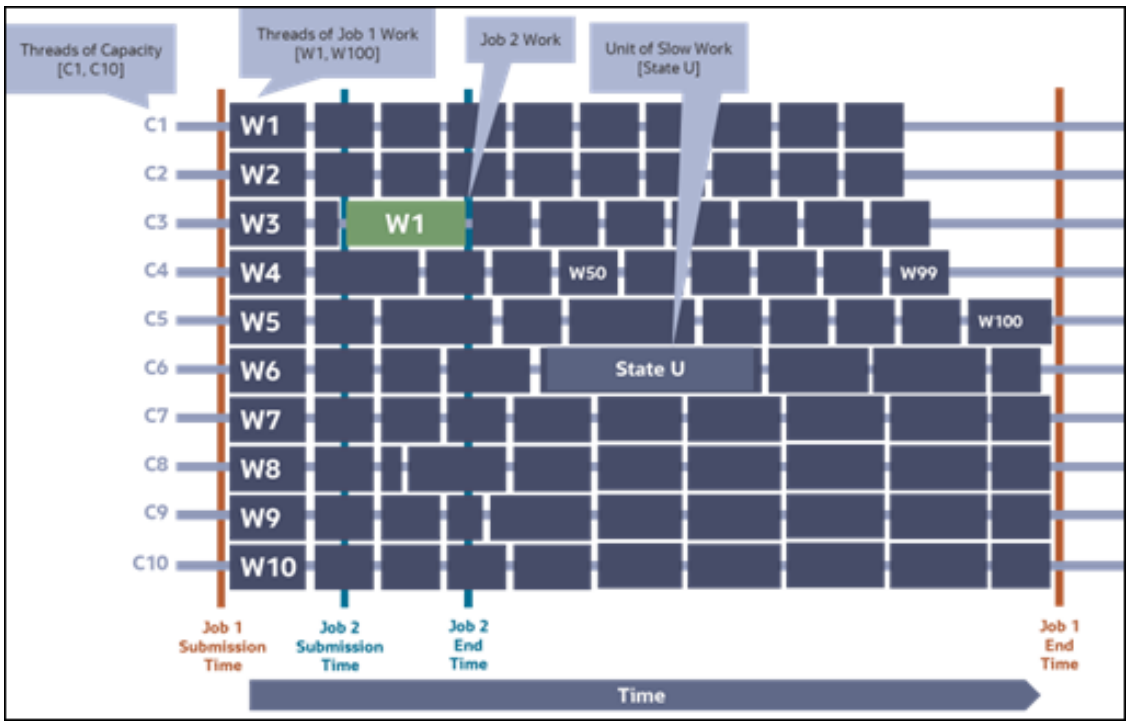
Fine-Grained Thread Submission

Technical features implemented in Oracle Utilities Cloud Services make possible the submission of jobs with much higher thread counts as well as more sophisticated policies governing how pending threads of work are matched to free threads of capacity. By higher thread counts we mean that a job may be submitted specifying far more threads of work than the number of threads of capacity available in the system. And since most of these threads of work cannot run immediately a queue of pending threads of work forms as do queues of pending work corresponding to other jobs similarly submitted. Having queued substantial threads of work, the system can employ concepts and algorithms of queue theory to solve the difficulties previously discussed:

- "Queue-based load leveling" lowers overall jobs' completion times even in cases of slow work items or capacity shortfalls.
- "Fair task scheduling" prevents resource starvation, fully utilizes capacity, and respects configurable processing priorities.

Let us get started by reimagining Job 1 being submitted with 10x more threads of work than described in the coarse-grained submission discussion. Instead of Figure 3-1 we would expect the following scheduling behavior:

Figure 3-4 Fine-Grained Submission



Solution #1: Delayed Job Completion from Thread Throughput Imbalances-Solved by Queue-based Load Leveling

Note in Figure 3-4 that Job 1's overall completion time is considerably shorter than in Figure 1. While the "State U" slow unit of work is still slow, its effect on overall completion time is

mitigated by the fact that the thread of capacity ("C6") on which the slow work is scheduled performs fewer threads of work than other threads. For example, while thread of capacity C6 completes 6 threads of work, thread of capacity C7 performs 11 threads of work. In essence, the well-behaved workflows around the poorly behaving work. Threads of capacity capable of performing more work receive more work to do while threads of capacity unable to complete work quickly are assigned less work. This behavior is the essence of what is known as "queue-based load leveling".

Note that is possible to construct scenarios where problematic work like "State U" happens to fall among the last threads of work scheduled (such as W100). However, it can be shown via statistical means that under most situations the queue-based load leveling shown here has far superior results than displayed by the coarse-grained thread scheduling.

Solution #2: Resource Starvation-Solved by Queue-based Load Leveling and Fair Task Scheduling

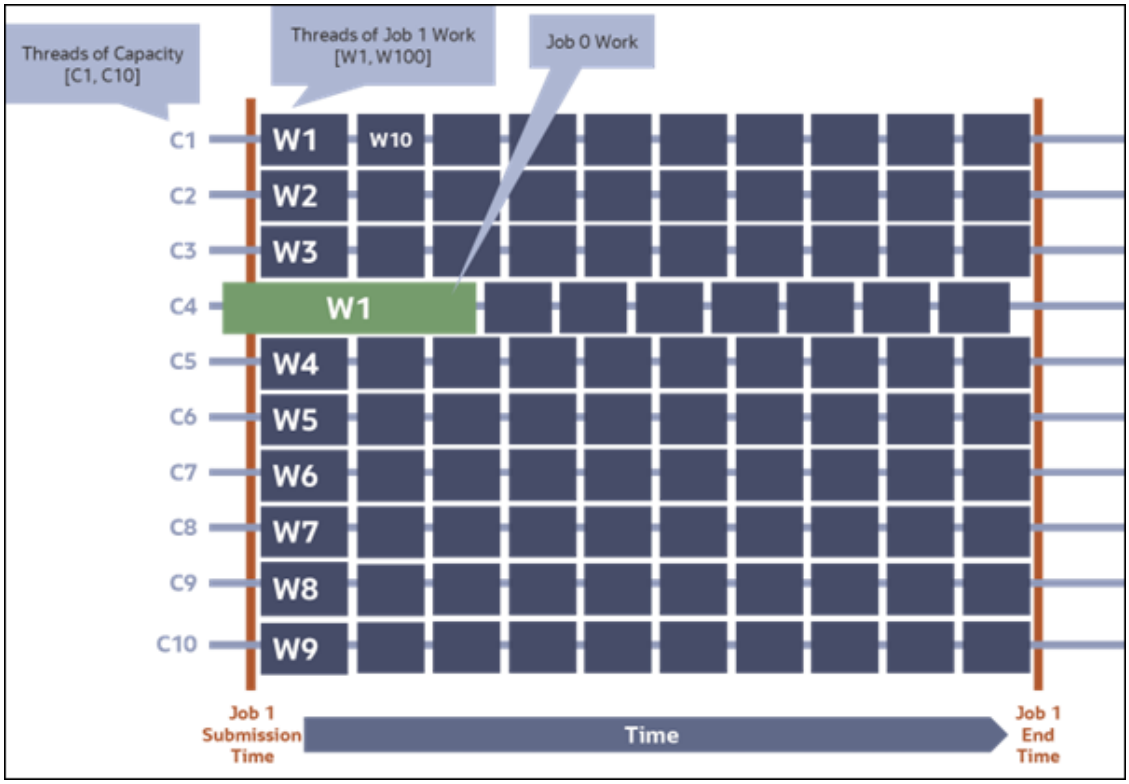
Note in Figure 3-4 that Job 2 starts processing and completes much sooner than in Figure 3-1. That is, we do not see Job 2 exhibiting resource starvation. This owes to two reasons:

1. Because of queue-based load leveling a thread of capacity becomes available to schedule Job 2 much earlier than in Figure 3-1. In figure 3-4 thread of capacity C3 completes two threads of work for Job 1 and then is available to schedule a thread of work from Job 2.
2. Because of fair task scheduling job 2 is guaranteed to receive some service (that is, have some threads of work scheduled onto threads of capacity) based on a fairness policy implemented in the batch processing framework. The fairness policy balances highly served jobs like Job 1 against minimally served jobs like Job 2. These policies are both powerful and configurable and will be discussed more in more detail later.

Solution #3: Delayed Job Completion from Thread Capacity Shortfall-Solved by Queue-based Load Leveling

Please consider Figure 3-5 and note that unlike Figure 3-2 that a thread of work from another job (Job 0) does not unreasonably extend the completion time for the newly scheduled Job 1. Rather, queue-based load leveling causes the impacted thread of capacity (C4) to preform less work (6 threads of work) relative to the other threads of capacity which process as many as 13 threads of work.

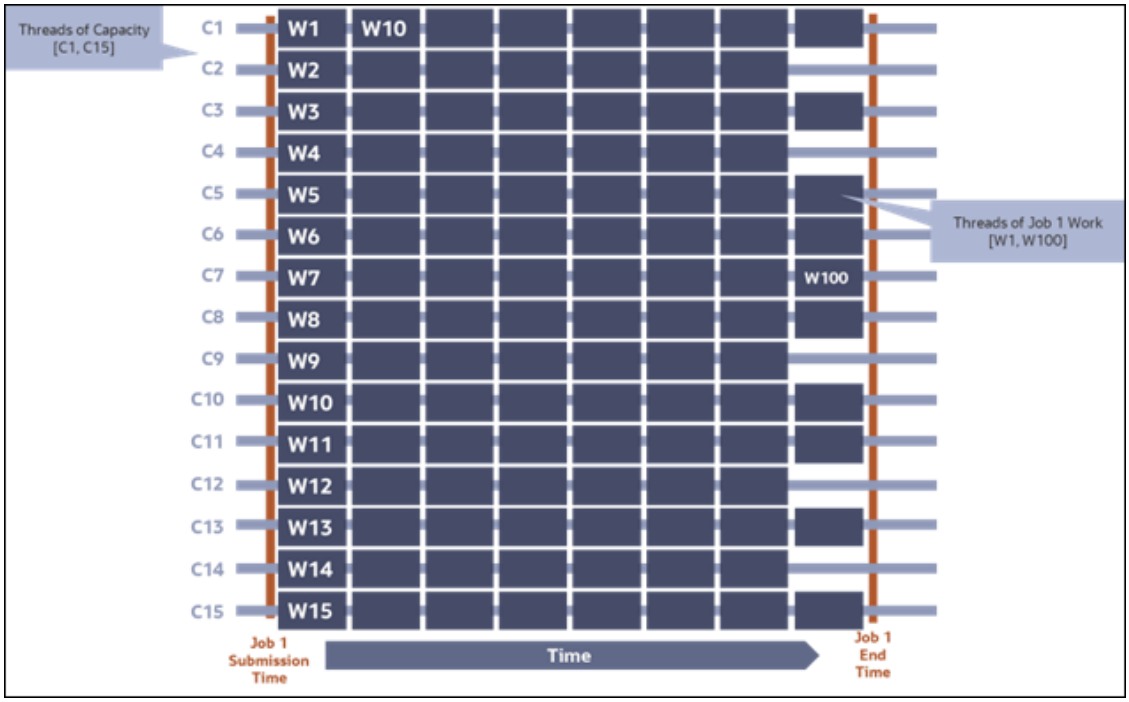
Figure 3-5 Fine-Grained Submission - Previously Running Job



Solution #4: Unused Capacity-Solved by Queue-based Load Leveling

It should be clear that the same property of queue-based load leveling which enables a job to handle shortfalls of thread capacity can also utilize excess capacity to further shorten completion job times. This is shown in Figure 3-6. When additional capacity is available it can be used to lower job completion times and if jobs are scheduled which require that additional capacity then the fair task queueing feature will ensure that those jobs receive threads of capacity according to the business-value optimizing policies defined through configuration.

Figure 3-6 Fine-Grained Submission - Excess Capacity Shortens Completion Times



Thread Count Recommendations

As described earlier, queue-based load leveling deals well with some threads of work having longer durations than the typical threads of work within a job. Nonetheless, there should be some well-defined maximum expected duration for typical threads of work. It also follows that this maximum thread duration should be targeted to be under 2 minutes. This prevents many threads of work from consuming threads of capacity for long durations and thus defeating the dynamic load balancing and fairness-based thread scheduling already been discussed. Thus, the recommendation offered here is to attempt to adjust the thread count of jobs such that the great majority of threads of work will finish in under two minutes.

Based on analysis of subscribers' batch processing within the cloud services the typical customer would need to increase their thread counts approximately 10x for 10-20 batch controls. The actual thread count achieving under-two-minute runtimes is expected to range somewhere between 200 and 1,000 threads of work. In general, only these 10-20 batch controls would need to be adjusted since all other jobs already complete in under 2 minutes or are single-threaded jobs. However, a very small number (one or two, typically) may need to have their thread counts increased by 40-100x, the "D1-IMD" batch control being a candidate for such an increase in thread count.

Fair Task Scheduling

As previously described, batch processing functionality has been enhanced such that jobs can be submitting with high thread counts which leads to many threads of work being initially queued by the system. Further, the system achieves business objectives like minimizing job completion times, maximizing resource utilization, and preventing resource starvation by optimally deciding which job to dequeue threads of pending work when multiple jobs have queued work. The common term denoting optimal choices like these is "fairness" and we will refer to the key objective in this context as "fair task scheduling". This section will describe the

cloud service's implementation of fair task scheduling and the ways that administrators can influence scheduling decisions to optimize business objectives.

Least Served Scheduling Among Flows

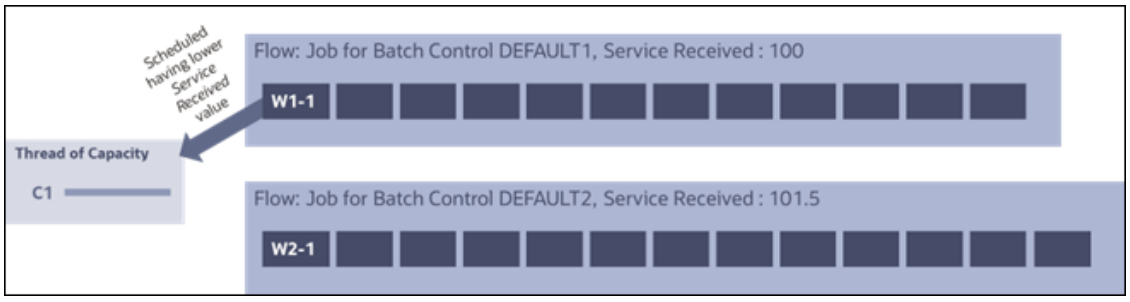
Much queueing theory originates from research into optimizing packet-switched networks and the terminology generally follows from that field. One term that we will use here is that of the "flow" which means a collection of backlogged work items that might be scheduled. A simple example of flows is where two batch jobs are submitted concurrently such that pending threads of work are queued for both jobs. In this case, each queue constitutes a flow, and the batch runtime must optimally choose to schedule threads of work for one flow versus the other.

The criteria for choosing one flow versus another is made based on tracking of the amount of "service" provided to each flow in the recent past. A flow that received less weighted service will be scheduled before another flow that has received more service. We will defer discussing the term "weighted" mentioned just now until the term "service" is better defined. Service is a scalar value representing the accumulated time that threads of work executed on threads of capacity. For example, if one thread of work executed for 30 seconds before completing then the flow that the work belongs would see its tally of service received increase by a value of 30. Likewise, if 5 threads from one flow each executed for 30 seconds then that flow's service received value would increase by 150.

Note that a thread of work does not need to complete for the service provided to be accumulated and therefore factor into scheduling decisions. Service quantities are incrementally applied to flows' tallies as threads of work run.

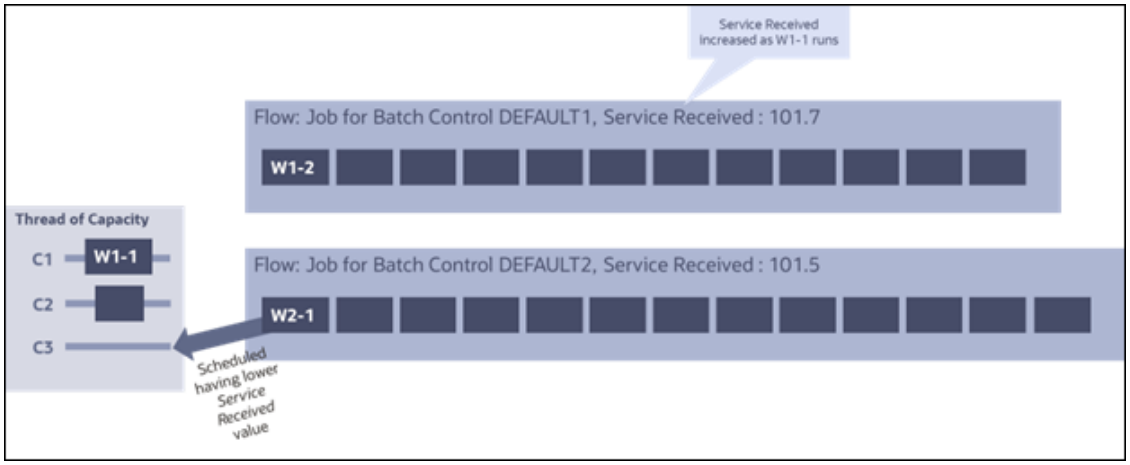
Consider Figure 3-7 below which shows that among two flows, batch jobs running batch controls DEFAULT1 and DEFAULT2, that a free thread of capacity is scheduled work from the flow having the lowest value of service received (100 vs. 101.5).

Figure 3-7 Two Flows



After some time, another thread of capacity becomes available to execute a thread of work. However, since work W1-1 from flow DEFAULT1 has been executing for some time, now flow DEFAULT2 has a lower value of service received. Therefore, work from flow DEFAULT2 is scheduled. Over time, each flow will receive approximately the same service because scheduling is performed from the flow having the lowest value of service received.

Figure 3-8 Two Flows - More Thread Capacity



Hierarchical Fair Scheduling Based on Resource Groups

Oracle Utilities Cloud Services enable extension of batch behavior by configuring Batch Resource Configuration extendable lookup values. One extendable lookup value should be specified for each Batch Control requiring non-default resource behavior. Specifically, a Resource Group value can be specified for each extendable lookup value and these Resource Group values serve two purposes in relation to fair queuing of batch work:

1. A Resource Group defines a higher-level flow wrapping the lower-level flows for the specific jobs as have already been discussed. These higher-level flows receive fair service relative to the flows of other resource groups.
2. A Resource Group defines a weight that governs how much service the resource group should receive relative to other resource groups. A higher weight receives proportionately higher service relative to a lower weight. That is, a Resource Group having double the weight of another resource group should receive approximately double the service over time.

The following table lists the delivered Resource Group values and their weights.

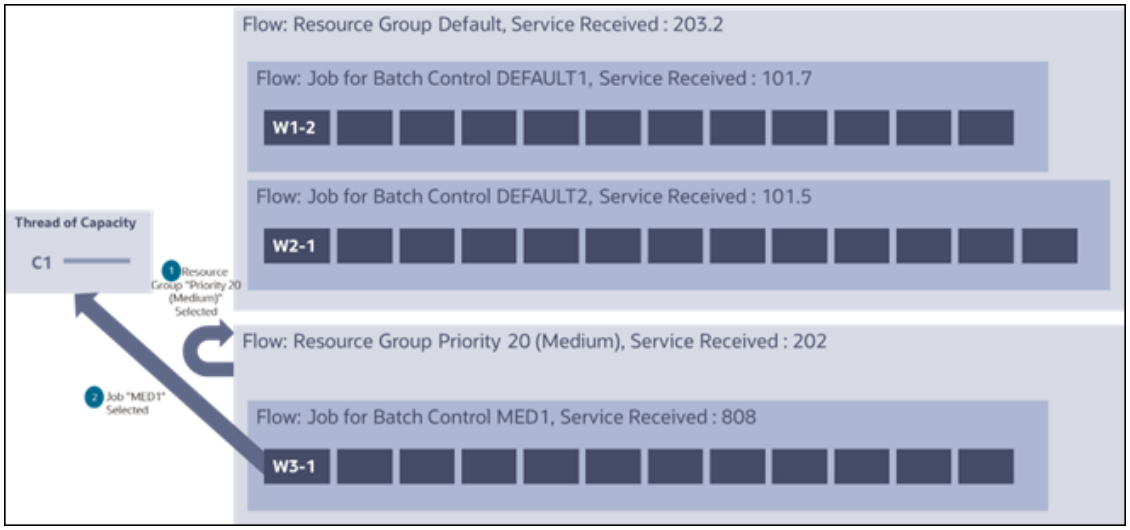
Resource Group	Weight
Default (not set)	1
Priority 30 (Low)	2
Priority 20 (Medium)	4
Priority 10 (High)	8

The idea that scheduling takes place via nested flows is commonly referred to as "hierarchical fair scheduling" and there are many examples of such algorithms in computation. In this case we define a two-level hierarchy where the higher level is represented by the resource group and the lower level by the individual jobs. Thus, scheduling of a thread of work to a thread of capacity occurs in two steps:

1. The resource group flow with the lowest value of service provided is chosen.
2. From the chosen resource group flow, the job flow with the lowest value of service is chosen.

Consider the following diagram showing three jobs competing for their backlogged threads of work to be scheduled. These jobs are of two different resource groups.

Figure 3-9 Three Jobs Competing for Backlogged Threads

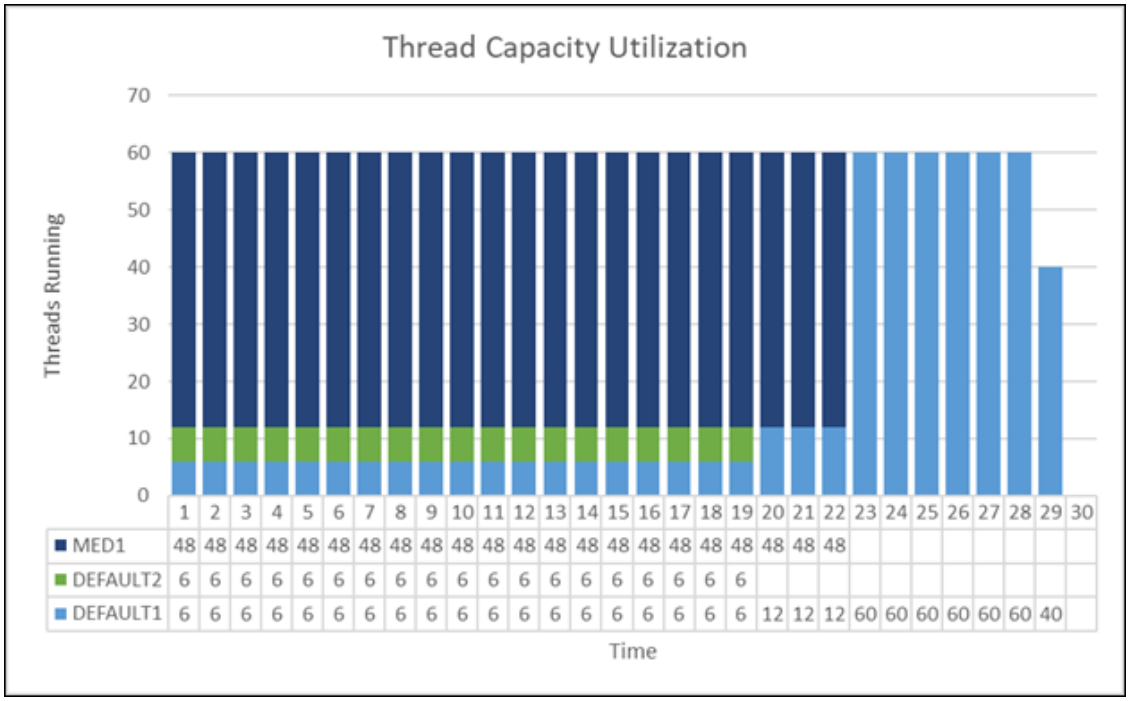


Note in the example that the value of service received for the Priority 20 (Medium) resource group is 202 while the value of service received by the job of batch control MED1 is 808. This is because the resource group flow increases its value of service received inversely proportional to its weight. In this case, the higher-level flow's value is 202 = 808 / 4 where 4 is the weight of the resource group. Thus, this resource group will receive four times the service over time relative to the Default resource group that has a weight of one.

The following figure (3-10) conveys how the three jobs might receive service over time assuming that the environment is entitled to 60 threads of capacity.

- While running, the MED1 job receives four time the service as the two other jobs combined (jobs DEFAULT1 and DEFAULT2) since the latter jobs are part of the default resource group but the former job (MED1) is part of the "Priority 20 (Medium)" resource group having four times greater weight.
- Within the default resource group, the DEFAULT1 and DEFAULT2 jobs receive equal service while they both run.
- When job DEFAULT2 finishes then the remaining job in the default resource group (DEFAULT1) receives all the service for the resource group which continues to be four times less that the "Priority 20 (Medium)" resource group job MED1.
- When MED1 completes then the remaining backlogged work for job DEFAULT1 utilizes all threads of capacity until completion.

Figure 3-10 Thread Capacity Utilization



Fair Queueing Considerations

System Convergence

The examples provided are idealizations of system behavior. However, actual results may deviate somewhat from what users might view as perfect fairness for various reasons. For example, because the history of service provided to a flow is used to make future scheduling systems there can be some system dynamical effects where the behavior is either over-damped or under-damped based on the composition and timing of jobs submitted over time. However, the system will converge to a state of approximate fairness over a reasonable timeframe.

Non-preemptive Work-conserving Scheduling

The hierarchical fair queueing algorithm described falls within the broad class of "non-preemptive work-conserving" scheduling algorithms. "Non-preemptive" means that once a thread of work is scheduled to a thread of capacity, the work will be allowed to complete even if higher priority work becomes queued and therefore cannot be immediately scheduled because of lack of free capacity. "Work-conserving" means that backlogged work will always be scheduled when free capacity exists. The combination of these choices promotes non-disruption of running processes and high resource utilization. However, the drawback is that submitting many threads of work where each thread takes a long time to complete might temporarily defeat the fair queueing algorithm by occupying most or all the thread capacity and therefore block the fair scheduling of subsequently submitted jobs.

The typical solution to the problem is to implement the fine-grained thread submission techniques already describe with particular attention to ensuring that most threads of work take no more than two minutes to complete. Thus, threads of work vacate threads of capacity often enough that overall scheduling fairness can be maintained by scheduling threads of work from under-served jobs onto the freed capacity. It is acknowledged that it may be impractically to

split some jobs, like file extracts, into many threads. However, in those cases leaving these extracts to run single-threaded or via a small number of threads is acceptable since these threads of work will only occupy a minority of threads of capacity at the same time and overall system fairness can be maintained by scheduling work onto the remaining capacity.

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**
>
> Migrating data between environments on different cloud service release versions using Content Migration Assistant (CMA), for example, importing data exported from a 25.10 environment into a 26.4Aenvironment, may result in import errors in some circumstances. Refer to **Versions and CMA** in the *Administrative User Guide* for more information about specific use cases where errors may occur.

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 Oracle Utilities System Health Check Probe (KB678481) on My Oracle Support 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.

Content Migration Assistant should 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
Number of concurrent reports	10
XML format report output	500MB
CSV format report output	1,000,000 Rows
Online/Browser report output	300MB
Scheduled (offline) report output	500MB
Database Storage	100GB

Attachment Limits

The size limit for attachments in the Oracle Utilities cloud services database is 25MB. Support is available for attachments larger than 25MB using Oracle Cloud Object Storage.

Batch Processing Limits

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

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

Business Log Processing Limits

Business logs are maintained per customer tenancy and are segregated for Staging and Prod partitions. Business logs are captured for batch and user logs (online and integration) separately. The retention period for business user logs is up to 15 days of data (30 days for batch logs) with a limit of 7,000 messages per minute. If your logs generate more than 7,000 messages per minute, this limit will be proportionally smaller to reflect the volume of data.

Identity Domain Limits

As part of your cloud service, the Identity Domain subscription type is **Oracle Apps Identity Domain**. Please refer to Limits BY Service in the Oracle Cloud Infrastructure documentation for limits associated with this domain subscription type.

OCI Email Delivery Limitations

The following limitations apply to OCI email delivery subscriptions included with your cloud service:

- Daily limit of 100 emails / Partition (prod/non-prod)
- Sending rate of 10 emails per minute
- 2 MB maximum message size including base64 encoding and headers
- Maximum of 10 recipients per email (in the TO:, CC:, or BCC: fields)

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 Database Actions (formerly known as SQL Developer Web):

Process	Limit
Database Actions 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	2 minutes	5 minutes*

* The timeout limit for web service calls can be extended up to 5 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.

> **Note**
>
> Using Inbound Web Services for data loading to your cloud service may have an impact on overall processing capacity and is not recommended.

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** portal for each thread via the **Download stdout** and **Download stderr** links.

Authorized administrators can also view tracing and debug logs of other users using 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).

Authorized administrators can also view and download access logs using the **Show Access Log** button when running the application with "debug=true" in the URL (see **Debug Mode** in the *Administrative User Guide*).

Outbound Allowlist Management

The customer or system integrator can request a DNS (Domain name service) name to be added in the allowlist for outbound interface communication. An allowlist provides access to specified DNS addresses that the Oracle network would otherwise prevent access to. For Oracle Utilities cloud services, a customer or system integrator must request a DNS to be added to the allowlist for outbound communication to all external systems.

There are multiple network access options supported, including:

- Accessing the endpoint DNS via public internet
- Accessing the endpoint DNS via Service Gateway (for accessing other oracle cloud services within th same region, such as Oracle Integration Cloud)
- Accessing the endpoint DNS via Private Endpoint (PE) or Reverse Connection Endpoint (RCE)

Once the requested DNS entry is added to the outbound allowlist, it is a customer responsibility to pro-actively maintain the following requirements:

- TLS / SSL Certificate should be issued by a valid SSL Authority
- Certificate's name(s) must match the server / endpoint name
- Installation of TLS / SSL Certificate should include complete authentication chain
- Expiry / Validation of TLS / SSL Certificate of the endpoint
- Support minimum of TLS 1.2

> **Note**
>
> Customers may use TLS / SSL validations tools such as openssl, TLS / SSL verification websites (https://www.ssllabs.com/ etc.) to validate the compliance requirements mentioned above.

Database Storage Details

Customer can view database storage details which include database storage size and high volume tables using Oracle Database Actions. This section provides SQL statements that can be used to view these details.

Note that the details generated via these SQL statements represent approximate database usage at a given point in time.

Database Storage - Total Size (in Terabytes)

Use the following SQL to view total database storage in terabytes (TB):

```
WITH my_nested_tables AS (
SELECT /*+ materialize */ owner, parent_table_name, table_name FROM
dba_nested_tables
WHERE owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN (
SELECT username FROM dba_users WHERE common = 'NO' AND oracle_maintained =
```

```
'N'))),
top_nested_tables AS (
SELECT /*+ materialize */ owner, parent_table parent_table_name, table_name
FROM
(SELECT owner, level, CONNECT_BY_ROOT parent_table_name parent_table,
table_name FROM my_nested_tables
CONNECT BY parent_table_name = PRIOR table_name) WHERE (owner, parent_table)
NOT IN (
SELECT owner, table_name FROM my_nested_tables))
SELECT SUM(bytes) / 1024 / 1024 / 1024 /1024 total_bytes_tb
FROM (
SELECT tab.owner, CAST('TABLE' AS VARCHAR2(11)) object_type,
CASE WHEN tab.iot_type = 'IOT_OVERFLOW' THEN tab.iot_name ELSE tab.table_name
END object_name,
CASE WHEN tab.iot_type = 'IOT_OVERFLOW' THEN tab.table_name END
auxiliary_name, nvl(seg_tab.bytes, 0) bytes,
nvl(tab.num_rows, 0) AS row_count FROM dba_tables tab
LEFT OUTER JOIN dba_segments seg_tab ON seg_tab.owner = tab.owner AND
seg_tab.segment_name = tab.table_name
WHERE tab.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN (SELECT username
FROM dba_users WHERE common = 'NO'
AND oracle_maintained = 'N')) AND nvl(tab.iot_type, 'NORMAL')!= 'IOT'
UNION ALL
SELECT ind.table_owner owner, CAST('INDEX' AS VARCHAR2(11)) object_type,
nvl(nst.parent_table_name, ind.table_name) table_name,
index_name auxiliary_name, nvl(seg_ind.bytes, 0) bytes, NULL AS row_count
FROM dba_indexes ind
LEFT OUTER JOIN top_nested_tables nst ON nst.owner = ind.table_owner AND
nst.table_name = ind.table_name
LEFT OUTER JOIN dba_segments seg_ind ON seg_ind.owner = ind.owner AND
seg_ind.segment_name = ind.index_name
WHERE ind.table_owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN ( SELECT
username FROM dba_users WHERE common = 'NO' AND oracle_maintained = 'N'))
UNION ALL
SELECT lob.owner, CAST('LOB' AS VARCHAR2(11)) object_type, lob.table_name,
lob.column_name auxiliary_name, nvl(seg_lob.bytes, 0) bytes, NULL AS row_count
FROM dba_lobs lob LEFT OUTER JOIN dba_segments seg_lob ON seg_lob.owner =
lob.owner AND seg_lob.segment_name = lob.segment_name
WHERE lob.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N'))
UNION ALL
SELECT nst.owner, CAST('NESTED' AS VARCHAR2(11)) object_type,
nst.parent_table_name object_name, table_name auxiliary_name,
nvl(seg_nst.bytes, 0) bytes,
NULL AS row_count FROM top_nested_tables nst LEFT OUTER JOIN dba_segments
seg_nst ON seg_nst.owner = nst.owner AND seg_nst.segment_name = nst.table_name
WHERE nst.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
```

```
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N'))
)
WHERE owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N'));
```

Top 100 Tables List by Size (in Gigabytes)

Use the following SQL to view the top 100 tables, listed by size in gigabytes (GB):

```
SELECT owner, object_name AS table_name, total_bytes_gb, row_count
FROM
(
WITH my_nested_tables AS (
SELECT /*+ materialize */ owner, parent_table_name, table_name FROM
dba_nested_tables
WHERE owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN (
SELECT username FROM dba_users WHERE common = 'NO' AND oracle_maintained =
'N'))),
top_nested_tables AS (
SELECT /*+ materialize */ owner, parent_table parent_table_name, table_name
FROM
(SELECT owner, level, CONNECT_BY_ROOT parent_table_name parent_table,
table_name FROM my_nested_tables
CONNECT BY parent_table_name = PRIOR table_name) WHERE (owner, parent_table)
NOT IN (
SELECT owner, table_name FROM my_nested_tables))
SELECT owner, object_name, SUM(bytes) / 1024 / 1024 / 1024 total_bytes_gb,
nvl(MAX(row_count), 0) AS row_count
FROM (
SELECT tab.owner, CAST('TABLE' AS VARCHAR2(11)) object_type,
CASE WHEN tab.iot_type = 'IOT_OVERFLOW' THEN tab.iot_name ELSE tab.table_name
END object_name,
CASE WHEN tab.iot_type = 'IOT_OVERFLOW' THEN tab.table_name END
auxiliary_name, nvl(seg_tab.bytes, 0) bytes,
nvl(tab.num_rows, 0) AS row_count FROM dba_tables tab
LEFT OUTER JOIN dba_segments seg_tab ON seg_tab.owner = tab.owner AND
seg_tab.segment_name = tab.table_name
WHERE tab.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN (SELECT username
FROM dba_users WHERE common = 'NO'
AND oracle_maintained = 'N')) AND nvl(tab.iot_type, 'NORMAL') != 'IOT'
UNION ALL
SELECT ind.table_owner owner, CAST('INDEX' AS VARCHAR2(11)) object_type,
nvl(nst.parent_table_name, ind.table_name) table_name,
index_name auxiliary_name, nvl(seg_ind.bytes, 0) bytes, NULL AS row_count
FROM dba_indexes ind
LEFT OUTER JOIN top_nested_tables nst ON nst.owner = ind.table_owner AND
nst.table_name = ind.table_name
LEFT OUTER JOIN dba_segments seg_ind ON seg_ind.owner = ind.owner AND
```

```
seg_ind.segment_name = ind.index_name
WHERE ind.table_owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%'
AND tablespace_name NOT IN ('SYSTEM', 'SYSAUX') AND owner IN ( SELECT
username FROM dba_users WHERE common = 'NO' AND oracle_maintained = 'N'))
UNION ALL
SELECT lob.owner, CAST('LOB' AS VARCHAR2(11)) object_type, lob.table_name,
lob.column_name auxiliary_name, nvl(seg_lob.bytes, 0) bytes, NULL AS row_count
FROM dba_lobs lob LEFT OUTER JOIN dba_segments seg_lob ON seg_lob.owner =
lob.owner AND seg_lob.segment_name = lob.segment_name
WHERE lob.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N'))
UNION ALL
SELECT nst.owner, CAST('NESTED' AS VARCHAR2(11)) object_type,
nst.parent_table_name object_name, table_name auxiliary_name,
nvl(seg_nst.bytes, 0) bytes,
NULL AS row_count FROM top_nested_tables nst LEFT OUTER JOIN dba_segments
seg_nst ON seg_nst.owner = nst.owner AND seg_nst.segment_name = nst.table_name
WHERE nst.owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N')))
WHERE owner IN (SELECT DISTINCT (owner) FROM dba_segments WHERE
tablespace_name NOT LIKE 'UNDO%' AND tablespace_name NOT IN ('SYSTEM',
'SYSAUX')
AND owner IN (SELECT username FROM dba_users WHERE common = 'NO' AND
oracle_maintained = 'N'))
GROUP BY owner, object_name ORDER BY SUM(bytes) DESC ) a
ORDER BY 3 DESC FETCH FIRST 100 ROWS ONLY;
```

4

Oracle Utilities Break Glass

Oracle Utilities Break Glass provides customers with transparency and control in terms of Oracle's access to data stored within Break Glass eligible Oracle Utilities Cloud Services, whether at rest or in use. This chapter provides guidelines for using Break Glass functionality, including:

Overview

Oracle Utilities Break Glass provides a means for customers (specifically, Customer administrators) to have direct control and approval authority over the process for requesting and granting temporary access to customer data by Oracle personnel.

Under normal circumstances, Oracle personnel are typically able to administer Oracle Utilities cloud service instances without access to customer data. In some cases, however, Oracle requires access to customer data to be able to perform certain important actions including (but not limited to): troubleshooting customer issues, applying certain categories of bug fixes, performing certain application upgrade steps or executing customer requested data correction activities.

For Oracle Utilities cloud service deployments that do not leverage Break Glass, Oracle personnel are granted temporary access to customer data via comprehensive internal controls and approvals based on SOC 1 and 2 principles. Oracle Utilities Break Glass plugs customer administrators into this process and gives them visibility and approval authority over all requests for access to customer data (by Oracle personnel).

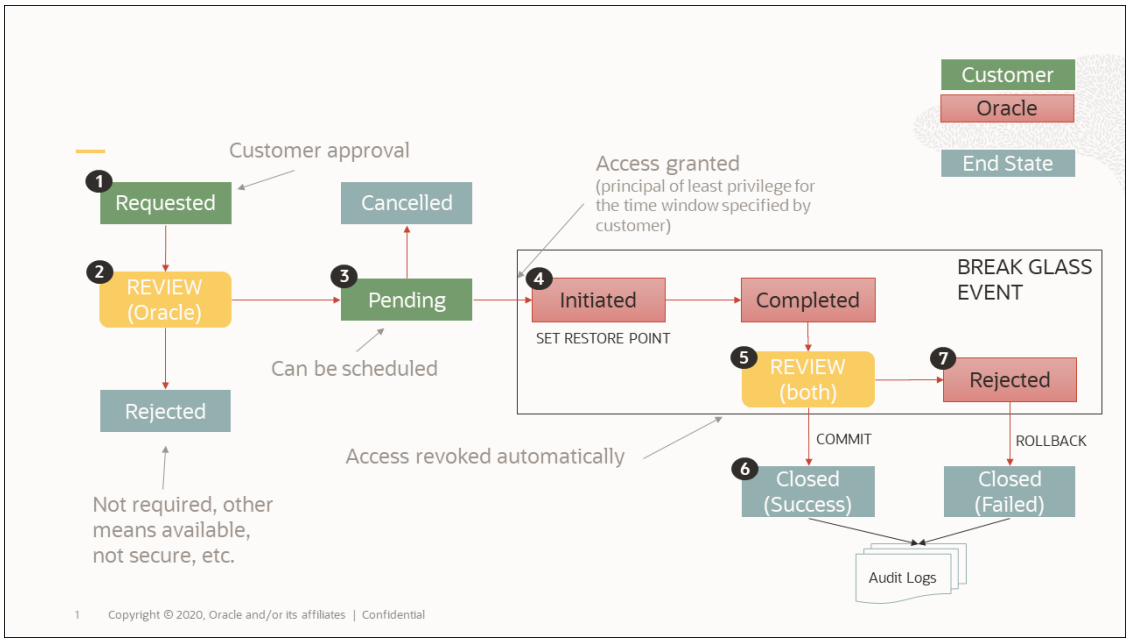
A Break Glass event defines a scheduled, time boxed process during which Customers provide approval for Oracle personnel to temporarily access their customer data to perform the requested actions on a specific tenancy and environment. Time boxing, in this context, means that Customers provide Oracle with access to their customer data for a limited period of time (defined in the Break Glass event) after which Oracle access is automatically revoked. The Break Glass event includes the following information:

- Customer, tenancy and environment (domain information)
- When the event should occur (scheduling), including planned start time and planned end time
- How long access to customer data is to be provided (time boxing)

- Which actions are required to be performed (by Oracle personnel) on customer data
- Description of what constitutes a successful outcome of the proposed actions
- Who requested the event
- Customer approvals
- Event and action statuses
- Audit log information

The following flow diagram outlines the general Break Glass work flow:

Figure 4-1 Break Glass Overview



1. Either Oracle or the customer requests a break glass event and includes details of schedule, duration (for example 24-96 hours) and detailed explanation of the activities to be performed by Oracle. The request process includes a customer approval workflow ensuring all requests will be reviewed/approved by customer representatives authorized to approve break glass activities.
2. Oracle reviews the request. Oracle may reject the request if, for example, Break Glass is not required (for example if other means of performing the activity are available) or if the request would pose a security risk (for example unsupported /destructive actions are requested)
3. Oracle schedules the request which awaits initiation in a pending status. Pending requests can be canceled.
4. Once Oracle has received and confirmed all required approvals are in place, Oracle initiates an automated process to provide the designated Oracle personnel with the minimum access credentials required to perform the task. These access credentials are provided for the time window established as part of the Break Glass Event. Oracle sets a restore point (to allow rollback to a safe/supported known state) and initiates the requested activities.
5. Once the requested activities are complete, both the customer and Oracle review the outcome, and determine if the activities were successful or otherwise.

6. Successful completion results in the changes being committed to the database
7. Failed completion results in the changes being rolled back to the established restore point

Upon closure of the Break Glass Event, the temporary access granted to Oracle is revoked and the audit logs are made available to the customer.

All communications relating to Break Glass Events are managed via My Oracle Support to ensure proper handling of potentially sensitive information.

Pre-Requisites for using Break Glass

The pre-requisites to using Oracle Utilities Break Glass functionality are as follows:

- Customers must be subscribed to a Break Glass eligible Oracle Utilities cloud service. Break Glass eligibility is explicitly specified in the service descriptions, and is currently available for 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 Customer Program Management Cloud Service
  - Oracle Utilities Market Settlements Management Cloud Service
  - Oracle Utilities Meter Solution Cloud Service
  - Oracle Utilities Rate Cloud Service
  - Oracle Utilities Work and Asset Cloud Service
- Customers must subscribe to the Break Glass cloud service add-on for each (eligible) Oracle Utilities Cloud Service for which they wish to use the Break Glass features. Oracle Utilities Break Glass is a fee-based service. Please contact your Sales representative to order this add-on service.
- Once subscribed, Customers may be required to raise a technical Service Request via My Oracle Support to enable Break Glass for a specific tenancy or environment. Please contact your Customer Success Manager for more information.

Establishing a Break Glass Event

Break Glass events are established by Oracle, but may be requested either by Oracle or by the Customer. For example:

- Oracle may identify a bug fix, upgrade step or troubleshooting activity which requires access to customer data. In this case, Oracle will establish the Break Glass event, specify the rational, propose an expected execution duration (i.e. the time box) and request that the Customer reviews/approves the requested access
- The customer may request that Oracle assist with troubleshooting an issue. In this case, the Customer would raise a Service Request (via My Oracle Support) to establish the Break Glass event. Please see the *Oracle Utilities Cloud Services Cloud Operations Guide* for details on how to raise this Service Request.

Break Glass events may be scheduled to happen as soon as possible, or they may be scheduled in the future.

Canceling a Break Glass Event

Any Break Glass event that has not yet been initiated may be canceled by Oracle or by the Customer. Break Glass events that have been initiated may not be canceled as Oracle must to

ensure that the cloud service is returned to a valid state. Customers may, however, request that any activities performed as part of the initiated Break Glass event (that have not already been committed) be rolled back, and this will be accommodated by Oracle where possible.

Approving a Break Glass Event

Customers must approve Break Glass events via My Oracle Support. This approval needs to be included in a Break Glass related Service Request, and must be made by a Customer administrator.

Reviewing the Results of Break Glass Actions

The results of actions performed during a Break Glass Event will be shared with the Customer to allow for review (together with Oracle).

The exact means by which the results will be shared (and discussed) will depend on the nature of the actions performed, and should be agreed to as acceptable by both Oracle and a Customer Administrator.

As review of Break Glass actions will occur during an initiated Break Glass event, Customer administrators (and any other required Customer personnel) must be available to review/ respond in a timely manner as the tenancy environment in question will be under scheduled downtime.

Please Note: the Customer administrator must ensure that all Customer personnel involved in the review of Break Glass action results are authorized to view those results, as the results may contain sensitive data.

Committing or Rolling Back Break Glass Activity

Once Oracle and the Customer have reviewed the results of the Break Glass activities, the changes must either be committed (perpetuated in the database) or rolled back (discarded).

While both Oracle and Customer personnel will be involved in reviewing the results of Break Glass activities, and while Oracle will work to provide relevant technical advice, it is the responsibility of a Customer administrator to make the decision to either commit or rollback any Break Glass actions performed.

Accessing Break Glass Audit Logs

Once a Break Glass Event is complete (either committed or rolled back), Oracle will provide an Audit Log of all activity performed by Oracle personnel on customer data as part of the Break Glass Event.

The audit log will be provided via a Service Request, will related to a specific Break Glass event, and will include details of any of the following Break Glass actions performed on customer data:

- Creation of any new customer data
- Reads of any customer data
- Updates to any customer data (including old and new values)
- Deletion of any existing customer data

Important Break Glass Considerations

- All Break Glass events will constitute scheduled downtime from the time the Break Glass event is initiated to when the actions performed are either rolled back or committed. Therefore:

  - Customer administrators must ensure timely response to all Oracle requests and timely involvement of customer personnel in reviewing Break Glass actions for the period of an initiated Break Glass event.
  - The System Availability SLO (Service Level Objective) does not apply during an initiated Break Glass event.
  - Uncommitted Break Glass actions may not be applied in the secondary instance in the event of a DR (Disaster Recovery) event occurring during a Break Glass event.
- Only customer administrators are able to approve or reject Break Glass events and related actions (for the customer).
- All approvals and sensitive data regarding Break Glass events must be provided to Oracle via My Oracle Support.
- Customer administrators are responsible for:
  - Ensuring that any non-Oracle personnel participating in Break Glass events or activity reviews are authorized to view customer data.
  - All decisions made regarding the committing or rolling back of Break Glass actions.
- For all customer initiated Break Glass events, the Customer is responsible for verifying the correctness of any requested changes.

5

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

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 Database Actions or Analytics Publisher to 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

Sequence	Parameter Name	Description	Detail Description	Required
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_idfrom d1_dvcwhere device_type_cd = :F1and d1_spr_cd = :F2and d1_device_id between :f1.lowId AND :f1.highIdorder 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.constants.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.

6

Information Lifecycle Management

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

- Information Lifecycle Management Overview
- Customer Responsibilities - Post Go Live

Information Lifecycle Management Overview

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 applying compression on existing partitions, running monitor processes to check archive data eligibility based on defined retention periods, copying the archive data to the cloud object storage, and when ready, dropping the partition.

Monthly database storage details which include database storage size and high volume tables can be viewed using Oracle Database Actions. See Database Storage Details for more information.

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

Customers are no longer required to run the Add Partition (K1-ILMAD) batch process every month. Oracle automatically adds partitions for entities, every month, on desired domains and schemas.

Customers will need to set up a batch stream for partition compression jobs for measurement tables (K1-ILMMC) and non-measurement tables (K1-ILMNC).Oracle recommends that customers should perform ILM testing in their Test environment before running ILM batch jobs in their Production environment.

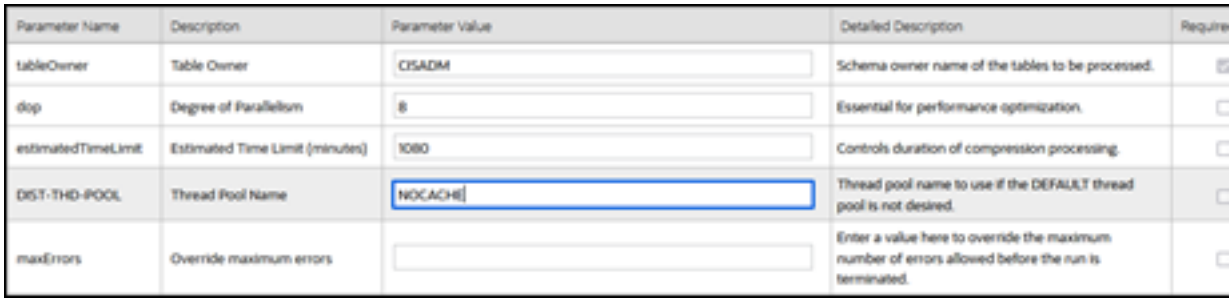
Exclusions: WACS supports ILM for F1 objects only (W1 prefix tables are not partitioned).

Customer Responsibilities - Post Go Live

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

- **ILM Automation - New Subretention Value** (F1-ILMSV): This process creates sub-partitions for Initial Measurements, Activities, and Device Events, based on specific sub-retention values defined in the ILM Configuration - MDM Sub-retention master configuration.
- **ILM Automation - Measurement Compression** (K1-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.

Figure 6-1 K1-ILMMC Parameters

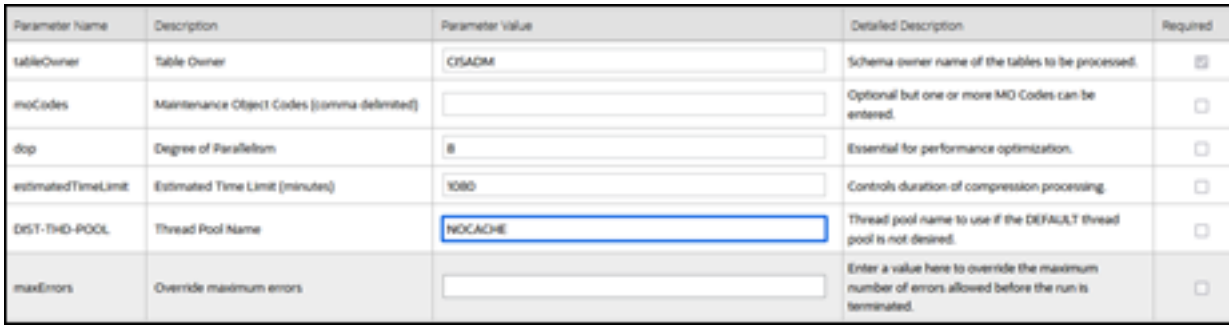


Parameter Name	Description	Parameter Value	Detailed Description	Require
tableOwner	Table Owner	CISADM	Schema owner name of the tables to be processed.	
dop	Degree of Parallelism	8	Essential for performance optimization.	
estimatedTimeLimit	Estimated Time Limit (minutes)	1080	Controls duration of compression 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 and during off peak hours.

- **ILM Automation - Non-Measurement Compression** (K1-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.
  If the K1-ILMNC batch process is submitted without the maintenance object parameter defined, it compresses all maintenance objects except D1-IMD. In order to compress D1-IMD, submit a separate batch job and specify D1-IMD as the Maintenance Object parameter.

Figure 6-2 K1-ILMNC Parameters



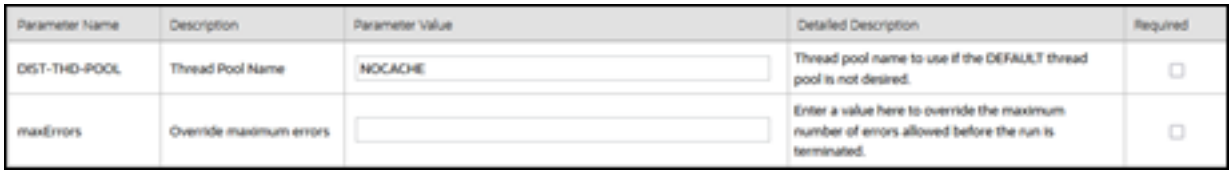
Parameter Name	Description	Parameter Value	Detailed Description	Required
tableOwner	Table Owner	CISADM	Schema owner name of the tables to be processed.	☑
moCodes	Maintenance Object Codes (comma delimited)		Optional but one or more MO Codes can be entered.	☐
dop	Degree of Parallelism	8	Essential for performance 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 and during off peak hours.

> **Note**
>
> ILM Compression jobs will run internally (like the ILM Add Partition job) by Oracle starting with the 26.10 release. Customers should start running both measurement and non-measurement compression jobs in any test-sized environment, followed by Production environments. Implementations can estimate the impact of compression on production environments by running compression processes in test environments. All ILM applicable table partitions MUST be compressed before upgrading to 26.10 in all Test and Production environments. Failure to do so may have performance impact to the online application following the upgrade to the 26.10 release.

- **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.

Figure 6-3 K1-ILMIN 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 and during off peak hours.

- **ILM Automation - Drop Partition** (K1-ILMDR): The ILM Automation - Drop Partition batch control provides the self-service capability to drop partitions for any ILM-applicable maintenance object without the need to raise a service request. Drop ILM partition support is also available for empty and old partitions for the all ILM-applicable maintenance object. Customers using previous releases should run the following batch process in sequence:

  1. Run the batch D1-IMDCL (ILM Crawler - IMD). This batch will update the ILM Archive Switch for drop partition eligibility. Guidelines for running this are as follows:
     - If the requirement is to retain the IMD data from more than 3 months (which is the default), make sure to override the default retention value on ILM Master configuration.
     - Don't run the batch with Override Cutoff Date batch parameter. Leave it blank to use current date so that batch should mark the IMD records based on the default retention value.
     - Batch supports multi-threading, so run it with maximum available threads.
     - Run the batch in the off peak hours.
     - Expect to run this batch longer (may be hours), depending upon the number of IMDs.
  2. Initiate a drop ILM partition process via the **ILM Dashboard** that runs the K1-ILMDR (ILM Automation - Drop Partition) batch internally, which supports one month partition drop at a time. Guidelines for running this are as follows:
     - Go to the **ILM Dashboard**. Click the **Partitions** tab.
     - Select the desired maintenance object and broadcast the tables and partitions details.
     - Click the filter on the **Partitions** zone and select "ILM Partition Range" filter to "Oldest 12 months".
     - Select the oldest partition (you can select one partition at a time and oldest partition to be selected as first).
     - Click **Drop Partition** (this creates a drop partition request that will go to the user who has been assigned an administrator role).
     - Once the request is approved by the administrator user, the K1-ILMDR (ILM Automation - Drop Partition) batch process will kick off and drop the desired partition.
     - The batch progress can be monitored using the **Batch Queue** portal or the **Batch Run** portal. The **ILM Dashboard also provide the request status such as in progress, error, completed, and so on.**
     - Run the batch in the off-peak hours.
  3. To drop the D1_MSRMT partition, customers are required to define the retention period on the Measurement maintenance object. The drop partition logic will first check the retention period on the Measurement maintenance object and if not configured, the drop partition job will look for a default retention period configured on the ILM Master Configuration.

  - If both configurations are missing, the drop partition job will fail to drop the oldest partition.

4. Raise a Service Request, in case of any issues and provide the following:
   - ILM Master Configuration showing default retention period and D1- IMD retention period.
   - Batch submission screen shots
   - Batch run screen shots
   - Both STDERR and STDOUT
5. Run these batches in any of your test environment first before running it in the Prod.

- Currently archiving is not supported. Only dropping existing partitions with or without data is fully supported.

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

7

Release Management and Regression Testing

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

Cloud Service Release Management

Oracle Utilities cloud services include software releases and updates, Infrastructure Updates, Maintenance Packs and Hot Fixes which Customers 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 two releases a year: April and October

- Releases introduce new features and may make improvements and changes to existing functionality
- The releases are labeled using the format YY.m (month). For example, the October 2025 release is called 25.10.
- Product documentation is made available in advance of each update.

Each release has a 'lifespan' of one year – in other words, 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 (for example, 25.10) 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 3 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 12)
- 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, 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:

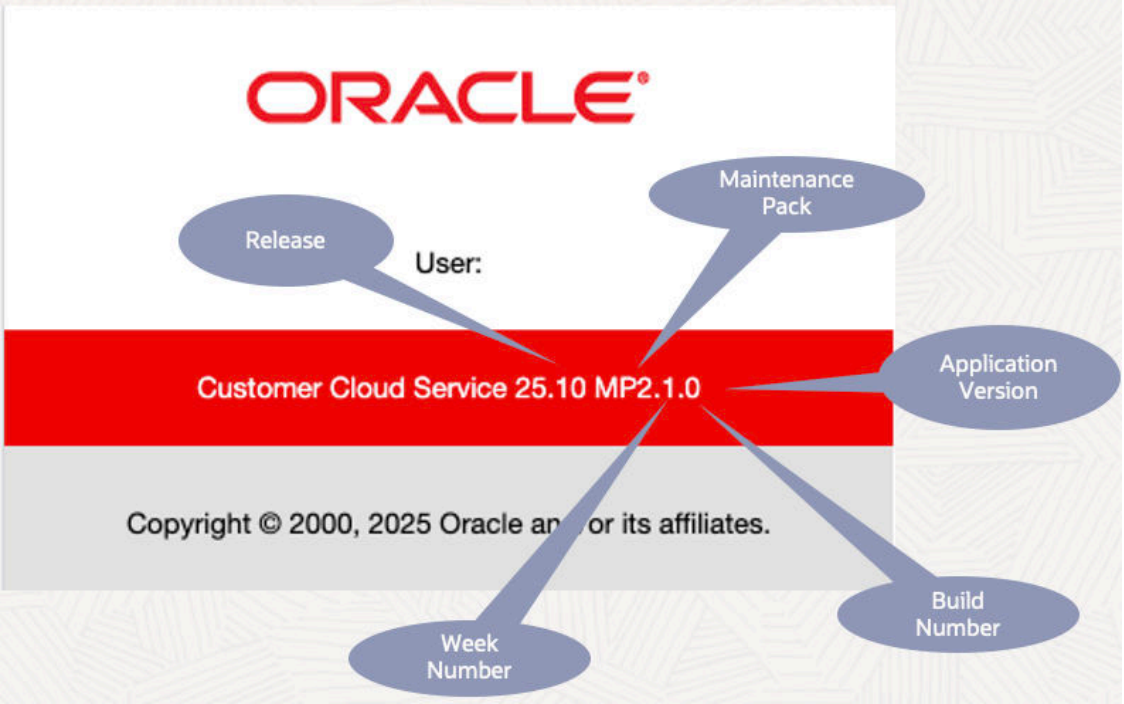
- Oracle recommends a 'three lane' approach to support customers thru the different stages of their lifecycle:
- **Implementation** or **Fast** lane: During most of the implementation project – early adoption of new releases as they become available
  - Uptake and use latest release with latest functionality, 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 6-month update cycle

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

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.

Figure 7-1 About Window



1. The "About" window displays data as outlined below (this example is for Customer Cloud Service 25.10 MP2.1.0):

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	25.10	MP2	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.

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 for Cloud Services documentation for more information.

Testing Assets

Testing Assets enable business process focused automated testing of Oracle Utilities cloud services using Oracle Utilities Testing Accelerator for Cloud, based on Oracle Utilities Reference Models. Testing assets are based on pre-configured administration and system data provided for Oracle Utilities cloud services (referred to as accelerator data).

Testing assets are available for the following Oracle Utilities cloud services:

- Oracle Utilities Customer Cloud Service
  Some Oracle Utilities Customer Cloud Service testing assets can also be used with:
  - Oracle Utilities Billing Cloud Service
  - Oracle Utilities Customer Care and Billing Cloud Service
  - Oracle Utilities Work and Asset Cloud Service
- Oracle Utilities Meter Solution Cloud Service

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.
- Review the **What's New** portal to view a list of application fixes applied to your environment as part any upgrade, such as hot fixes, maintenance packs, or release upgrades.

Customers should create regression testing plans based on the changes to your environment listed in the **What's New** portal.

- Customers taking new releases are required to perform the following steps:
  - Review the Oracle Cloud Application Update Readiness portal.
    - Review the What's New document for your release.
    - 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** portal and resolve errors in custom code, if found.
  - Review the **Application Services** section in What's New release readiness document. Determine if your implementation wants to opt in to any of the functionality related to the new features secured by the listed application services, and grant security access to the appropriate users.
  - 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.

Function-Specific Regression Testing Recommendations

Oracle recommends testing the following functional areas of the following cloud services as part of regression testing with each release:

- Oracle Utilities Customer Care and Billing Cloud Service
  - Bill Print
  - Start/Stop
  - Payment Integration
- Oracle Utilities Customer Cloud Service

  - Bill Print
  - Start/Stop
  - Payment Integration
  - Smart Meter Commands
- Oracle Utilities Meter Solution Cloud Service
  - Smart Meter Commands

8

Disaster Recovery

This chapter describes Disaster Recovery with regard to Oracle Utilities Cloud Services, including:

What is Disaster Recovery?

Disaster Recovery (DR) is the planning and implementation of processes to support the fail-over of specific services to an alternative location in the event of a disaster impacting the primary deployment location, typically (and in the case of Oracle) at the data center / regional level (see below for further definition).

The term "disaster" is formally defined in section 3 "ORACLE CLOUD SERVICE CONTINUITY POLICY" of the Oracle Industries Cloud Services Pillar Document. Without prejudice to that document (or to your legal agreement with Oracle), a "disaster" means an unplanned event or condition that causes a complete loss of access to the primary site used to provide the Industries Cloud Services such that Your production environments at the primary site are not available.

Disaster Recovery vs High Availability

High Availability (HA) is a feature of a deployment architecture which provides *resiliency* against common component failures and faults, typically through the use of hardware and resource redundancy (for example where "spare" resources are deployed in the primary region in case active resources fail).

A reasonable summary of the relationship between Disaster Recovery and High Availability is as follows:

- High Availability is designed to mitigate component failures/faults (e.g. to servers, compute, memory, routers, switches, storage/disks, etc.) *within* the primary region or availability domain
- Disaster Recovery is designed to address the failure of the primary region or availability domain *as a whole*.

What is hopefully clear is that both High Availability *and* Disaster Recovery need to be implemented to provide a satisfactory level of overall resiliency and redundancy.

Why do I need Disaster Recovery?

Disaster Recovery provides contingency against large scale events (known as "disasters") which cause a complete loss of access to the primary site.

While it is not possible to list all potential scenarios that could result in the loss of the primary site, common examples include:

- Natural disasters (earthquakes, storms, floods, hurricanes, and so on)
- Data center failures (fire, flooding, electricity, telecommunications, cooling, and so on)

How does Disaster Recovery Work?

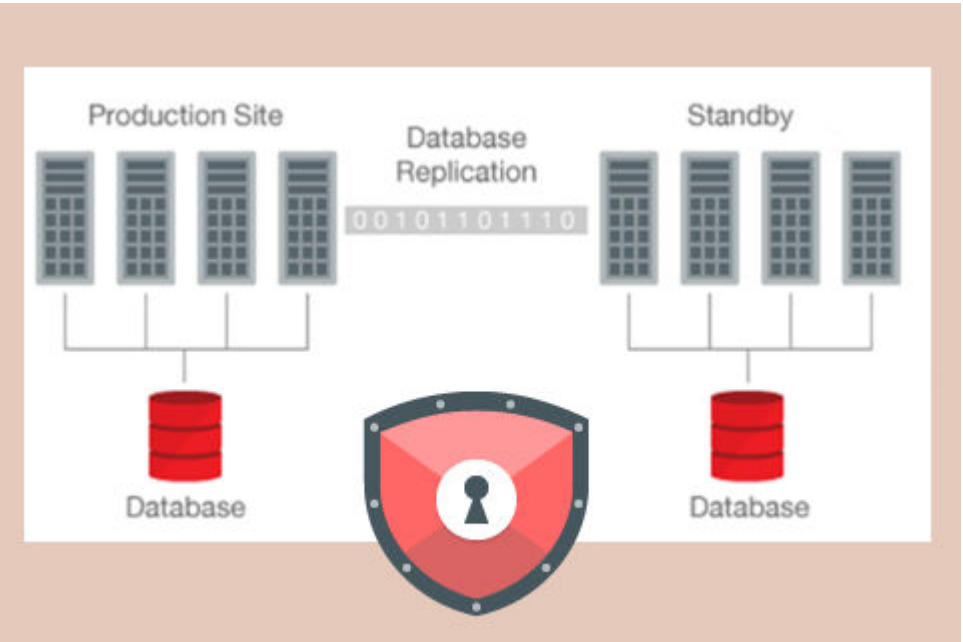
Oracle Utilities Enterprise SaaS solutions include full disaster recovery for the Production environment.

This means that a separate secondary (or fail-over) data center is provided in addition to the primary data center, such that:

1. Fail-over hardware for production is available on standby.
2. Transactions and data in the production primary database are securely replicated to the secondary database.

Under normal operating conditions, Active Data Guard replicates changes made in the primary database instance to the secondary database instance on a transactional basis (in other words. as changes are committed to the database).

Figure 8-1 Disaster Recovery - Database Replication



These Disaster Recovery capabilities are of particular benefit to utilities, as reliable disaster recovery is often very challenging to set up and maintain, particularly when large databases are involved (for example more than 5TB with high rates of data change).

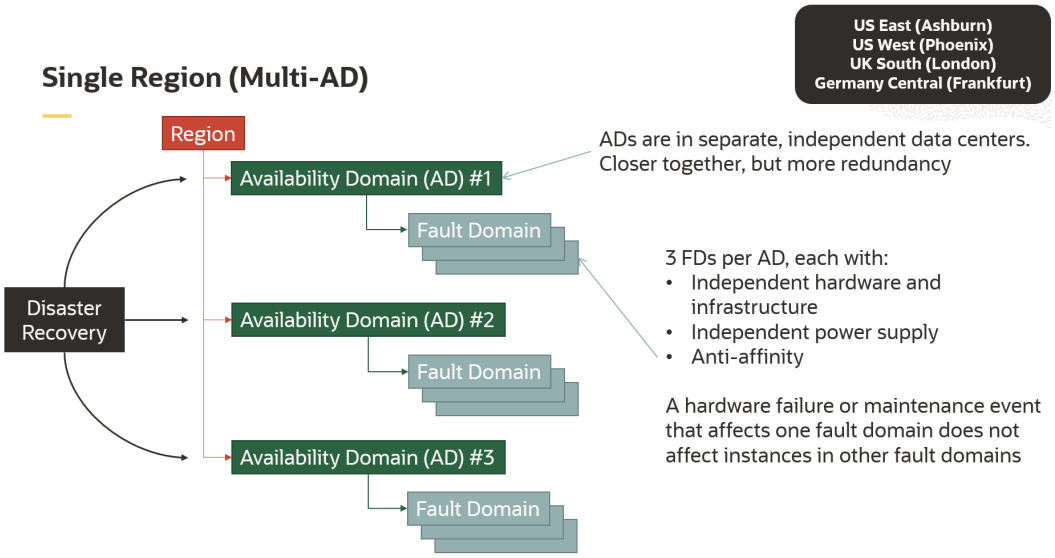
Oracle Cloud Infrastructure (OCI) is deployed in Regions and Availability Domains (which include Fault Domains).

- A Region is defined as a localized geographic area.
- An Availability Domain (AD) consists of one or more data centers within a region
- A Fault Domain (FD) consists of independent hardware, infrastructure and power within an availability domain.

Oracle operates two general types of OCI Regions:

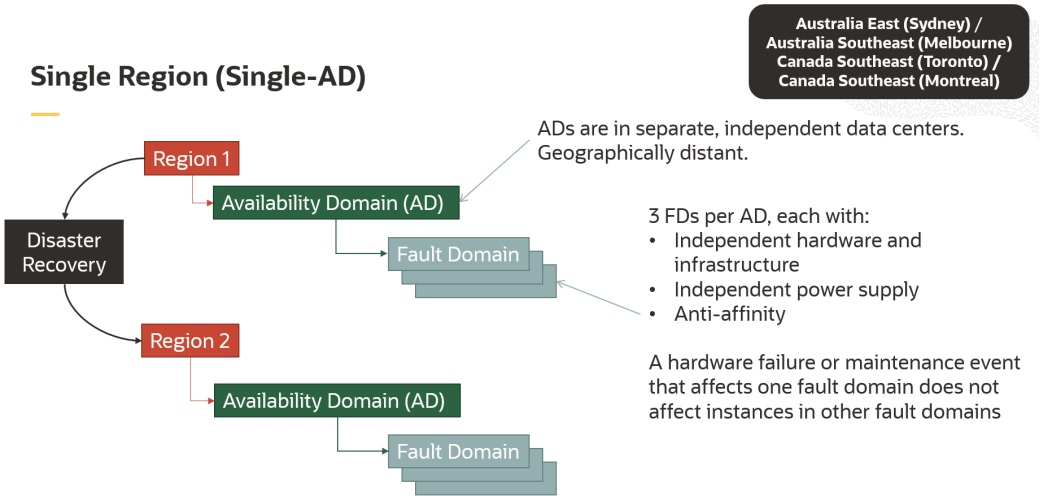
- Single Region (Multi-AD) regions with three (3) ADs each. One AD is the primary, with the other two being available as secondaries.

Figure 8-2 Disaster Recovery - Single Region (Multi-AD)



- Single Region (Single-AD) regions with one (1) AD each. In this case, two regions are paired (called a region pair) where Region 1 is the primary and Region 2 is the secondary.

Figure 8-3 Disaster Recovery - Single Region (Single-AD)



Oracle Utilities Enterprise SaaS solutions are available in the following regions:

- US East (Ashburn), US West (Phoenix), UK South (London) and Germany Central (Frankfurt) which are Single Region (Multi-AD) regions.
- Australia East (Sydney) Primary / Australia Southeast (Melbourne) Secondary and Canada Southeast (Toronto) Primary / Canada Southeast (Montreal) Secondary which are Single Region (Single-AD) region pairs.

> **Note**
>
> For Single Region (Single-AD) region pairs, the primary/secondary arrangement is fixed and alternative configurations are not currently supported.

Please note the following, additional Disaster Recovery considerations:

- Disaster Recovery failover is not available (for example for business continuity testing) on a customer specific basis (as failover is executed at the region/data center level).
- Oracle Identity Domain capabilities are also included in Disaster Recovery failover.
- All hardware required for operation of the protected environments is reserved in the secondary.
- To ensure seamless and transparent failover, FQDNs (Fully Qualified Domain Names) should always be used.
- In some regions, customers will need to configure replication for their subscribed Object Storage containers (if required).
- Disaster Recovery is specific to the subscribed cloud services (where applicable) and does not extend to cover 3rd party systems or integrations. These need to be covered by customer-managed business continuity planning.
- For operational integrity and security reasons, the secondary database is not accessible by customers unless a real-world disaster recovery fail-over has occurred, and service is restored in the secondary location.

Disaster Recovery Coverage

Disaster Recovery is only formally provided for the Production environment.

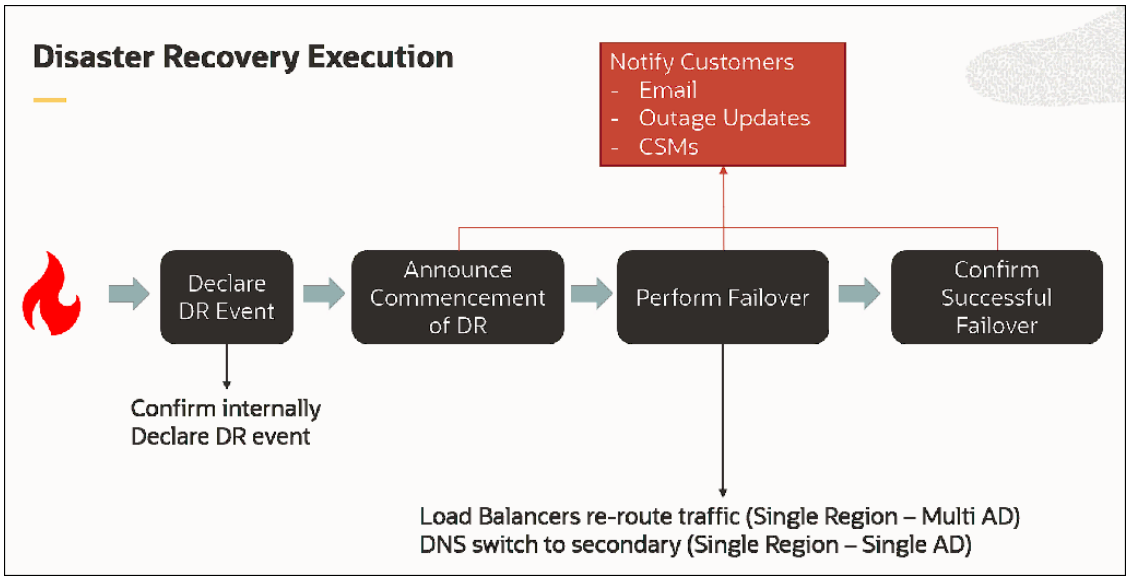
Some non-production environments may also be protected (meaning they are included in DR), however the related DR SLOs (RTO/RPO) only apply for cloud services and environments where this is explicitly stated in the Service Descriptions.

Disaster Recovery Execution

Oracle Cloud Infrastructure has a public facing page with current status and information on previous incidents. It can be found here: https://ocistatus.oraclecloud.com/

Customers are notified of a disaster declaration and pending failover by the Cloud Operations team and Service Delivery Managers, as outlined in the following diagram.

Figure 8-4 Disaster Recovery Execution



Bearing in mind that a declared disaster is likely impacting all customers with deployments in the affected OCI Region(s), Oracle will use reasonable efforts to:

- Update impacted customers as per a Severity 1 (Critical Outage) as defined in the Cloud Hosting and Delivery Policies,
- Provide regular restoration time frame estimates.

The decision is made via a human workflow according to a predefined processes/SOPs/ standard Oracle policy, and involves executive Oracle management.

For security and privacy reasons, Oracle does not disclose details of this human workflow.

Oracle will work with you to return service to your primary data center as soon as possible.

Disaster Recovery Testing

Per Oracle Corporate Security Practices (specifically Risk Management Resiliency Disaster Recovery) Oracle information technology organizations conduct an annual Disaster Recovery exercise (or more frequently as required, for example when the applications, services or tools are updated) for internal infrastructure designed to assess Disaster Recovery plans. Lessons learned from the exercise are implemented into standard operations and Disaster Recovery procedures as appropriate.

Disaster Recovery plans and tests are reviewed and updated based on experience and feedback, and the teams responsible for Disaster Recovery responsibilities are provided with regular training. Testing is performed on non-production tenancies and not on production customer databases.

Disaster Recovery Evidence Summary Reports may be made available to existing cloud service customers upon request.

Role of a Cloud Customer in Disaster Recovery Testing

As Oracle cloud service Disaster Recovery testing is conducted at a regional level, it is not currently possible to support customer participation in these testing events.

Furthermore, Disaster Recovery of Oracle Utilities cloud services is limited in scope to those cloud services only and is largely transparent to the end customer. For these reasons, customer participation in such testing (as performed by Oracle) would be of limited value.

Cross-Region Disaster Recovery Preparation

In order to prepare for disaster recovery (including cross-region disaster recovery), customers are required to perform the following:

- Firewall Update: Customers accessing cloud applications via their home region are required to get standby region cluster IP addresses (via a My Oracle Support Service Request) and allow the standby cluster IP addresses in their on-premises firewall configurations.
- Object Storage: Refer to **Cross-Region Disaster Recovery Considerations** in the *Oracle Utilities Cloud Services Administration Guide*.
- Identity Domain Syncing: Refer to the guidelines in the Disaster Recovery and Identity Domains section in the Oracle Cloud Infrastructure documentation.
- OCI Network Configuration: Customer must review their OCI network configuration, including Service gateway (any services created within their tenancy for incoming and/or outgoing integrations).

9

GoldenGate Replication

This chapter provides information about using Oracle GoldenGate Replication with Oracle Utilities cloud services, including:

GoldenGate Replication Overview

GoldenGate Replication provides the ability to establish one-way, initial, and ongoing change-based (Change Data Capture) replication of data from an Oracle Energy and Water Software-as-a-Service (SaaS) cloud service environment (for example, a Customer Cloud Service Production environment) to an OCI hosted Autonomous Database Autonomous Transaction Processing (ATP) target environment (owned and managed by the customer) via a Named Distribution Path.

This provides you with full and direct access to all SaaS data replicated to the target Autonomous Database ATP database instance, and enables you to query, report on, and develop data-driven integrations based on your cloud service data in the target database. It also allows for retention of data that is useful for reporting and integration purposes, even if the data is removed from the source cloud service database via ILM processes such as partition drops.

Refer to the Oracle Cloud Infrastructure GoldenGate and Oracle GoldenGate documentation for more information.

Refer to the **GoldenGate Replication** chapter of the *Oracle Utilities Cloud Services Administration Guide* for more information, including requirements, prerequisites, and instructions for administration tasks used with GoldenGate Replication.

Using GoldenGate Replication

Once connectivity is established between your cloud service tenancy and your target tenancy, you can request a database copy of the source environment. This database copy is then transferred to your object storage location in Oracle Data Pump format, from where your database administrators can then import the database copy into your target Autonomous Transaction Processing database instance.

There are important considerations regarding the initial load process:

- Initial load extract files can be large, and are directly proportional to the source database size. Information Lifecycle Management (ILM) should be used in the source database to ensure it is as small as possible.
- The extract may require downtime in the source database (approx. 1 hour) to establish proper synchronization data required for replication, and therefore needs to be scheduled during a scheduled maintenance window (or will be considered scheduled maintenance).
- Full schema extract is used to maximize performance.
- Target database autoscaling should be enabled during the initial load.

Assuming the Database Copy is successfully imported into the target Autonomous Transaction Processing database, you can then configure your OCI GoldenGate service to begin applying Change Data Capture (CDC) information received via the Named Distribution Path to your target database instance in order to keep the target synchronized with the source.

Also, note the following important operational considerations:

- Ongoing replication is not the full schema, as the following are excluded:
  - High change working tables (for example, to track batch execution)
  - IMD CLOBs (MSCS, AMS)
  - The staging schema
  - Custom tables (via Java Migration Cloud Service) until data volumes are assessed)
- Multiple target replicats are recommended to prioritize/optimize replication performance. It is recommended that groupings are based on:
  - Data volumes
  - Priority
  - Table parent/child relationships
- Lag after heavy processing the source (for example, nightly batch processing) should be expected
  - Prioritized and parallelized replicats should catch up within several hours if properly sized.
- Upgrades and Data Model Changes:
  - DDL is largely excluded except CREATE and ALTER TABLE statements to add new tables/columns
  - Indexes, partitions, and more may be created via initial load, but are not maintained via replication (ILM partition drops are not replicated in the target, as customers are likely to want to keep that data)
  - You are responsible for managing any required changes to the target database in terms of partitioning, indexes, and so on, as these will be highly dependent on the reporting and integration requirements.
- Retention of trail files is 7 days
  - This allows recovery within that time period
- Re-establishment of replication services is possible, and will involve:
  - A new full schema data extract/initial load
  - Restarting of replicats
  - Data management to merge the new source data with existing target data

10 Troubleshooting

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

- Database Monitoring Using Database Actions
- Running and Troubleshooting Batch Processing

Database Monitoring Using Database Actions

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 Database Actions (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.

Database Actions (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.

> **Note**
>
> The intended audience of the Performance Hub report are database administrators with experience in extracting, interpreting, analyzing database performance information.

Running the Performance Hub Report

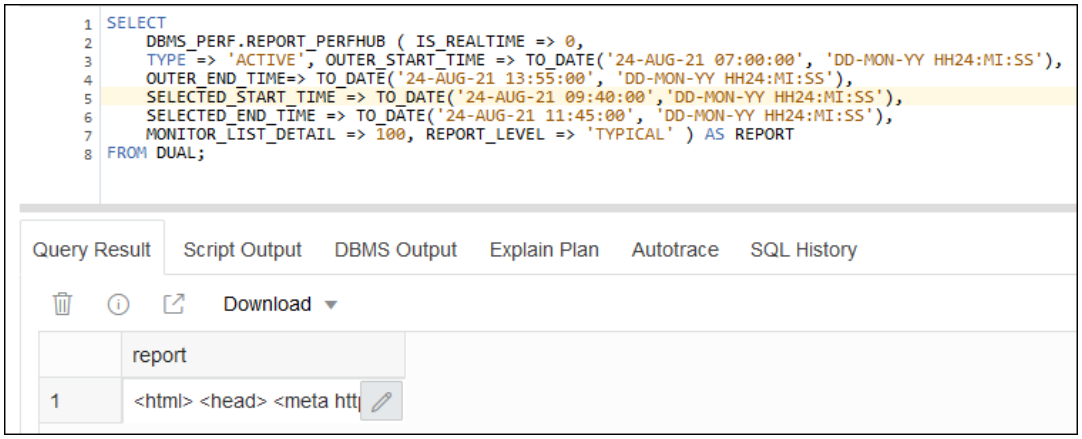
Use the following procedure to generate the Performance Hub Report:

1. Using Database Actions (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 Database Actions:

Figure 10-1 Performance Hub Report - Database Actions SQL Statement



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:

Parameter	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)
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

Figure 10-2 Sample Output - Snippet 1

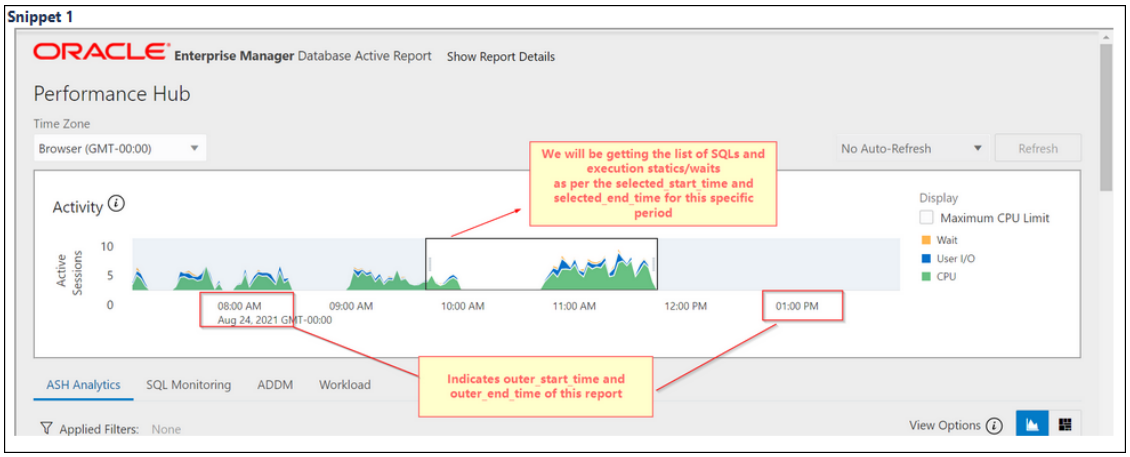
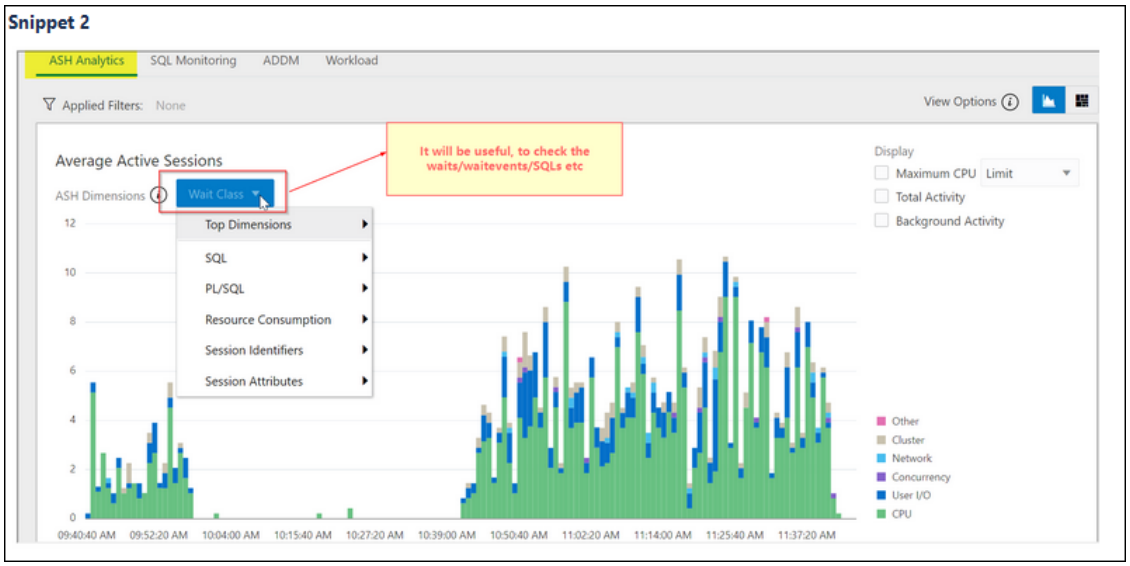


Figure 10-3 Sample Output - Snippet 2



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 Database Actions (ORDS) Explain Plan and AutoTrace functionality. The following screen shots illustration these functions.

Figure 10-4 Explain Plan

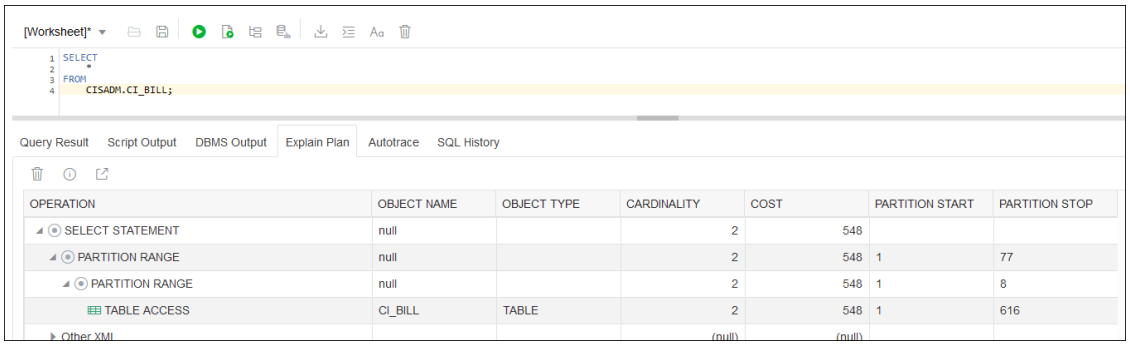
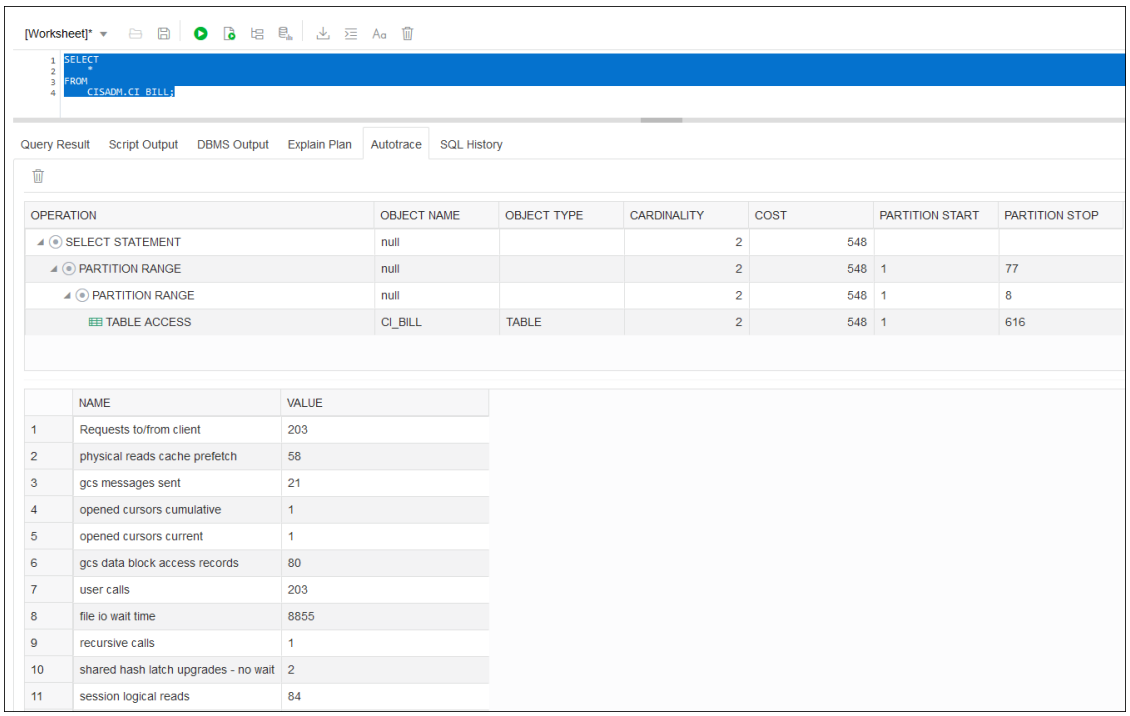


Figure 10-5 AutoTrace



Running and Troubleshooting Batch Processing

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

Running and Troubleshooting Batch Processing - 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.

Running and Troubleshooting Batch Processing - 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 K1-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** portal.

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 - in other words, 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.

Running and Troubleshooting 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 (in other words, 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, K1-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.

Running and Troubleshooting Batch Processing - 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.

Running and Troubleshooting Batch Processing - Frequently Asked Questions for Common Batch Issues

Q1) Batch job results in error, what should I do?

First step is review the **Batch Run** portal. The **Batch Run** portal 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 looking at the thread step on the **Batch Run** portal. In some cases the thread details will show the error message that was thrown and that may be enough to understand the problem.

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 1:

Figure 10-6 Batch Log Error

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 2:

The **Batch Run** portal 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
```

> **Note**
>
> The above example illustrates a connectivity issue. In cases like this, please raise a Service Request and provide screenshots of the batch job submission, with parameters, from the **Batch Run** portal and both logs (STDERR and STDOUT). For batch logs, Oracle maintains logs data for 14 days only, so we recommend that you download them promptly.

Please note that a batch run may complete successfully even though certain records may have encountered recoverable errors (and in certain cases may create To Do entries for follow-up). So particularly in early batch testing it is important to examine the **Batch Run** portal 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 the batch are marked as DO NOT ATTEMPT RESTART.
- Re-submit the batch job with single thread and check the **Trace SQL** and **Trace Output** boxes on the batch job submission
- Once the batch job results in error, go to the **Batch Run** portal 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, from the **Batch Run** portal, and both logs (STDERR and STDOUT).

> **Note**
>
> If a custom-built batch errors, please contact the project implementation / support team.

Q2) After a batch results in error and after re-submission, why does the Batch Run portal shows the same batch number and Batch Run portal 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** portal, and click **Do Not Attempt Restart**.

Q3) The Batch Run portal 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 a Service Request and the Cloud Operations team will need to review technical logs to find the underlying error.

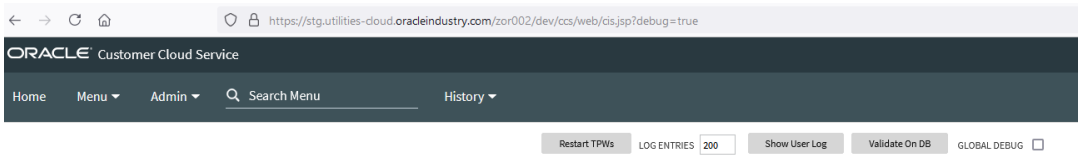
Q4) I have submitted batch online and the batch job submission still shows batch status in Pending?

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).

Figure 10-7 Debug Mode



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

Q5) I have canceled 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.

Figure 10-8 Batch Queues List

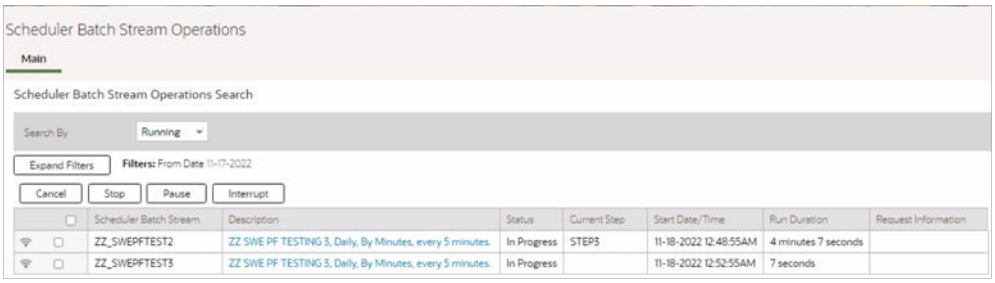
Batch Queues List

	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) How do I cancel jobs submitted by SaaS Batch Scheduler?

A Scheduler Batch 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 canceled through the cancel option on batch job submission page. In order to cancel the Scheduler Batch Stream, you have to go to **Scheduler Batch Stream Operations** portal page, then select the running stream and perform cancel operation. This will result in canceling the overall stream including current running batches.

Figure 10-9 Scheduler Batch Stream Operations portal



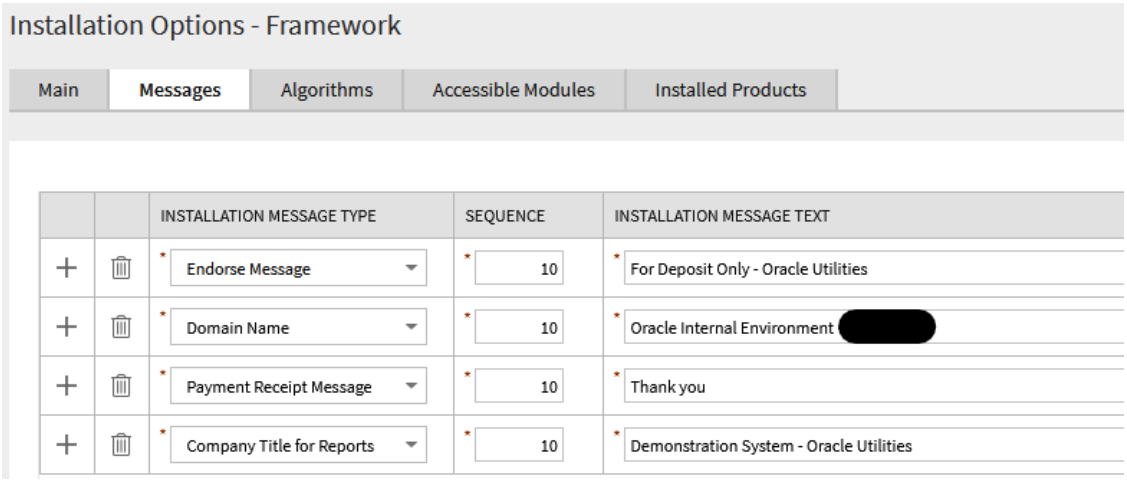
Q8) Why do I see multiple instances of F1-BTMON batch in batch job submission?

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?

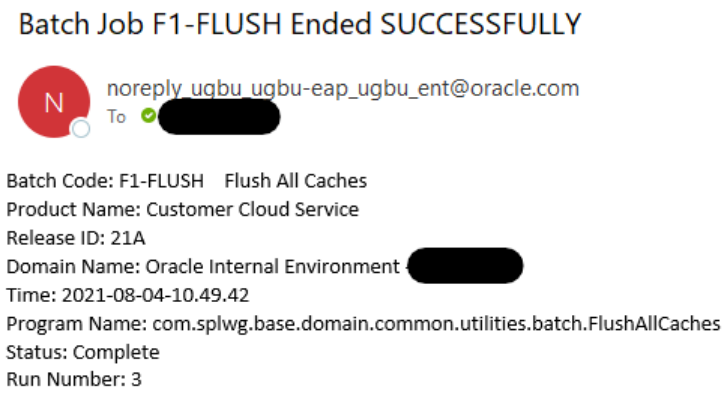
Yes, implementations can add a "Domain Name" Installation Message type (on the **Messages** tab of the **Installation Options-Framework** portal) and provide the description that identifies the environment. That description will be part of the email received as part of batch completion as well as on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

Figure 10-10 Installation Options - Framework portal



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

Figure 10-11 F1-FLUSH Batch Job



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

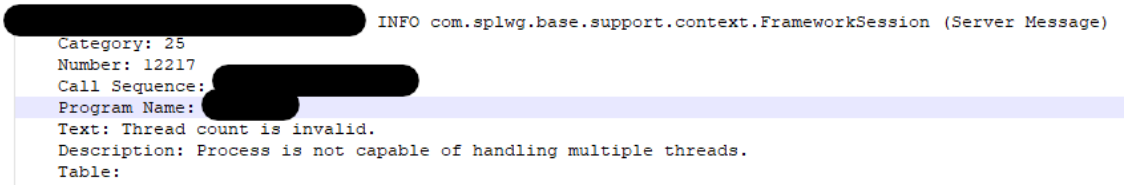
Q10) Can I run any job with many threads?

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:

Figure 10-12 Batch Log



Q11) Batch job submission shows 'Started' but the Batch Run portal 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** portal, to restart against what's left to process.

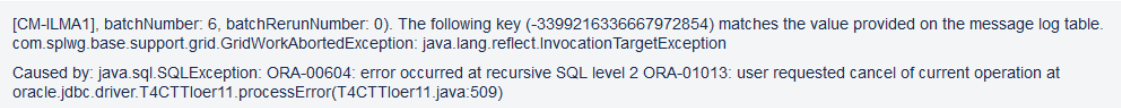
Q13) The logs for my batch job are empty - what should I do?

There are a handful of jobs that will not produce logs - these are mostly related to ILM processing (K1-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.

Figure 10-13 Sample Log - Timeout



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.

Q18) Where can I find information about handling SaaS Batch Scheduler exceptions?

Refer to **Handling Exceptions** in the **Scheduler Batch Stream Operations** section of the *Oracle Utilities Cloud Service Foundation Administrative User Guide* for guidelines related to handling exceptions using the SaaS Batch Scheduler.

Q19) How do I know which batch thread is processing which records?

In order to identify the records being processed by a particular thread, run the following SQL statement using Oracle Database Actions:

```
SELECT ROWNUM THREAD_NBR
, LPAD(ROUND((ROWNUM - 1) * (RPAD('9',:KEYLEN,'9')
+1)/:THREADS,0),:KEYLEN,'0') LOW_ID
, LPAD(ROUND(ROWNUM * ((RPAD('9',:KEYLEN,'9')+1)/:THREADS),0)-1,:KEYLEN,'0')
HIGH_ID
FROM DUAL
CONNECT BY ROWNUM <= :THREADS;
```

Parameters:

- :THREADS = Number of threads used on batch job submission
- :KEYLEN = The number of digits as key length

The output of this SQL statement will list all running threads and their corresponding ranges. This will help you identify which thread is running specific records.

Q20) Some of my batch threads are stuck (not processing any records). How can I find out which records those stuck threads are processing?

In order to identify the records being processed by a stuck thread, find the batch process number using the **Batch Run** portal, look for the stuck thread number to find the record’s ID (this is the last ID successfully processed by the batch thread). Find the next record ID by using the above SQL statement, and use this ID to process the record manually using the application and look for any abnormality.

Glossary

Index