



Teradata® Unity™

Monitoring and Management Guide

Release 17.00




September 2020

Copyright and Trademarks

Copyright © 2020 by Teradata. All Rights Reserved.

All copyrights and trademarks used in Teradata documentation are the property of their respective owners. For more information, see [Trademark Information](#).

Product Safety

Safety type	Description
	Indicates a situation which, if not avoided, could result in damage to property, such as to equipment or data, but not related to personal injury.
	Indicates a hazardous situation which, if not avoided, could result in minor or moderate personal injury.
	Indicates a hazardous situation which, if not avoided, could result in death or serious personal injury.

Third-Party Materials

Non-Teradata (i.e., third-party) sites, documents or communications ("Third-party Materials") may be accessed or accessible (e.g., linked or posted) in or in connection with a Teradata site, document or communication. Such Third-party Materials are provided for your convenience only and do not imply any endorsement of any third party by Teradata or any endorsement of Teradata by such third party. Teradata is not responsible for the accuracy of any content contained within such Third-party Materials, which are provided on an "AS IS" basis by Teradata. Such third party is solely and directly responsible for its sites, documents and communications and any harm they may cause you or others.

Warranty Disclaimer

Except as may be provided in a separate written agreement with Teradata or required by applicable law, the information available from the Teradata Documentation website or contained in Teradata information products is provided on an "as-is" basis, without warranty of any kind, either express or implied, including the implied warranties of merchantability, fitness for a particular purpose, or noninfringement.

The information available from the Teradata Documentation website or contained in Teradata information products may contain references or cross-references to features, functions, products, or services that are not announced or available in your country. Such references do not imply that Teradata Corporation intends to announce such features, functions, products, or services in your country. Please consult your local Teradata Corporation representative for those features, functions, products, or services available in your country.

The information available from the Teradata Documentation website or contained in Teradata information products may be changed or updated by Teradata at any time without notice. Teradata may also make changes in the products or services described in this information at any time without notice.

Machine-Assisted Translation

Certain materials on this website have been translated using machine-assisted translation software/tools. Machine-assisted translations of any materials into languages other than English are intended solely as a convenience to the non-English-reading users and are not legally binding. Anybody relying on such information does so at his or her own risk. No automated translation is perfect nor is it intended to replace human translators. Teradata does not make any promises, assurances, or guarantees as to the accuracy of the machine-assisted translations provided. Teradata accepts no responsibility and shall not be liable for any damage or issues that may result from using such translations. Users are reminded to use the English contents.

Feedback

To maintain the quality of our products and services, e-mail your comments on the accuracy, clarity, organization, and value of this document to: docs@teradata.com.

Any comments or materials (collectively referred to as "Feedback") sent to Teradata Corporation will be deemed nonconfidential. Without any payment or other obligation of any kind and without any restriction of any kind, Teradata and its affiliates are hereby free to (1) reproduce, distribute, provide access to, publish, transmit, publicly display, publicly perform, and create derivative works of, the Feedback, (2) use any ideas, concepts, know-how, and techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, and marketing products and services incorporating the Feedback, and (3) authorize others to do any or all of the above.

Contents

Chapter 1: Alerts and Errors Management Overview	4
Considerations during Unity Setup	4
Basic Maintenance Concepts	6
Chapter 2: Unity Alerts	9
Recommended Alerts	9
Setting Up Alerts	10
Checking Alerts	11
Clearing Alerts	11
Chapter 3: Manual Monitoring	12
Checking Unity System Health	12
Checking DFS Space Usage	13
Checking HA Sync Status	13
Checking Managed System Health	14
Checking Dispatcher HA	15
Checking Object Health	15
Checking for New TVI Messages	17
Chapter 4: Information from Log and Configuration Files	18
Unity Log Files	18
Unity Diagnostic Files	18
/var/log/messages	18
Unity Configuration Files	18
Unity Support Script	19
Chapter 5: Unity Troubleshooting	20
Interrupted Objects	20
Unrecoverable Objects	24
Critical Events	25

Alerts and Errors Management Overview

Unity enables multiple active Teradata systems to work as one analytical ecosystem to manage the following:

- Data synchronization
- Database changes
- Query routing
- High-priority alert messaging using the Notification Service that is bundled with the Unity UI

This guide provides key concepts and instructions for alerts, monitoring, maintenance, and errors. For more information, see the *Teradata® Unity™ User Guide*.

Teradata Vantage™ is our flagship analytic platform offering, which evolved from our industry-leading Teradata® Database. Until references in content are updated to reflect this change, the term Teradata Database is synonymous with Teradata Vantage.

Advanced SQL Engine (was NewSQL Engine) is a core capability of Teradata Vantage, based on our best-in-class Teradata Database. Advanced SQL refers to the ability to run advanced analytic functions beyond that of standard SQL.

Considerations during Unity Setup

Unity uses monitoring rules to send critical status conditions to administrators. The monitoring rules are part of the Notification Service that is bundled with the Unity UI.

For best practices, do the following:

- Determine use case, including routing and system modes.
- Collect site information.
- Set up the recommended alerts.
- Perform regular maintenance and save health check output.
- Monitor for errors.
- Respond to alerts.

Determining Use Case

Routing rules specify the Teradata system and how read and write requests are routed.

Before setting up alerts, review the Unity routing and system mode use case considerations.

Unity Routing Modes

Routing Mode	Description
Passive Routing	<ul style="list-style-type: none"> • Use Passive Routing to provide session-based routing when you do not want data synchronization. • System selection is based on routing rules. • No data synchronization occurs and objects are not listed as managed by Unity in object and system status. • Passive Routing can provide session failover in the event of system outages.
Managed Routing	<ul style="list-style-type: none"> • Use Managed Routing when you need data synchronization. • Unity sends write statements to managed systems to make sure all managed objects remain in sync. • Managed routing setups may have passive-routing users. • Objects used by managed routing are not writable by passive users.

Unity System Modes

System Mode	Description
HA pair	<ul style="list-style-type: none"> • Standard Unity operation mode. • Unity has one Active and one Standby sequencer, running on two different servers. • Systems are kept in peer to allow for failover of the sequencer in the event of Active sequencer failure.
Standalone	<ul style="list-style-type: none"> • Unity has a single sequencer running on a single server. • Sequencer failover is not possible. • Not used on production systems, except for passive-routing-only configurations.

Collecting Site Information

Make note of the following Unity site information when contacting Teradata Customer Support:

- Site IDs for Unity servers
- Site IDs for the Managed systems and the corresponding Unity system ID
- Unity routing mode
- Unity system mode

Saving Health Check Output

Analyzing Unity trends over time may be useful. Some issues may resolve on their own over time and other issues are expected based on the current setup.

Document key metrics so trends can be seen over time. Key metrics include the following:

- Object counts
- Object states
- Recovery log usage
- Data directory space
- Alert counts

Monitoring Alerts

Like any critical system architecture, Unity needs monitoring rules in place to send critical status conditions to administrators. Unity uses the Notification Service that is bundled with the Unity UI to send notifications.

For optimum performance, do the following:

- Set up the recommended alerts.
- Perform regular maintenance.
- Monitor for errors.
- Respond to alerts.

Basic Maintenance Concepts

Unity routes queries to one or more managed systems that are independent Teradata database systems using one of the following methods:

Routing Method	Description
Passive routing	Unity sends each query to a single system as defined by routing rules.
Managed routing	Unity sends read queries to a single system and sends write queries to all TD systems managing that object.

With managed routing, if an error occurs or a system falls behind, Unity replays missed transactions to maintain synchronized data. To maintain a Unity cluster and make sure all components are working, you must monitor all managed systems and the Unity system.

To ensure overall system health, review and apply the following maintenance concepts:

Concept or Condition	Description	Maintenance Action
Unity Management User	Unity uses a special database user called the Unity Management User on all managed systems. The Unity Management User is essential to all Unity operation.	Make sure the Unity Management User is not obstructed by database throttles.
Alert monitoring setup to push critical events	Unity needs emergency notifications. Several Unity alerts indicate a critical condition that needs to be resolved.	<ul style="list-style-type: none"> • See Recommended Alerts. • Resolve alerts as quickly as possible to maintain operation.

Concept or Condition	Description	Maintenance Action
Raised alerts	Unity raises alerts for system conditions that need to be addressed.	<ul style="list-style-type: none"> • Check alerts regularly. • Fix and close valid alerts, then close any alert that is not closed automatically. • Disable alerts that are raised as a result of normal system operation. For example, when the default routing rule is expected to be used, disable the "Default Routing Rule used" alert, if it is acceptable to use the default routing rule.
Interrupted objects and systems	When an error or other condition on a system prevents a query from being run, objects and systems may become interrupted. Some interrupts resolve on their own automatically, others require intervention. If there are interrupted objects, other objects may become interrupted	Repair interrupted objects and systems as soon as possible. Fix the underlying error to make the object active again. See Manual Monitoring .
Unrecoverable objects and systems	When a data mismatch occurs, Unity makes objects unrecoverable. That may cause additional objects to become unrecoverable.	Use logs to determine the cause of the data mismatch, then fix the underlying issue. Resynchronize unrecoverable objects and systems as soon as possible to restore dual-active service.
Data resynchronization plan	You must synchronize data after a data mismatch or after objects and workloads are added.	Use a utility such as Data Mover or another duplication strategy to synchronize individual objects or entire systems.
Non-Unity connections	For Unity to replicate data, run SQL through Unity instead of connecting directly to the managed system. Connecting directly to managed systems may cause mismatched data, unexpected deadlocks, and object or system interrupts due to resource contention.	<p>If there are unexpected data mismatches or deadlocks, check DBQL to make sure objects are not being accessed from outside Unity.</p> <p>Resync unrecoverable objects and systems as soon as possible to restore dual-active service.</p>
Sessions managed from outside Unity	Unity manages multiple sessions to multiple managed systems. If a session is lost, Unity attempts to reconnect the session.	If you need to kill a session associated with Unity, kill it at either unityadmin or the Unity UI so that Unity knows that session is not to be reconnected.
Mismatched system capabilities	When one managed system has significantly greater processing power than the other managed systems, objects may become interrupted if the system with less processing power is unable to handle the load of the queries.	Make sure all managed systems can handle workload for both space and processing capacity. If users run queries beyond the capability of a system, consider passive routing for the user or set the job objects to be managed only on the capable system.

Concept or Condition	Description	Maintenance Action
TASM restrictions	If TASM rules are mismatched, objects may become interrupted when queries are sent to all systems and aborted on only one system. If load slots are mismatched, loads to all systems are constrained by the system with fewer load slots.	Make sure all managed systems have matching TASM rules for all users and jobs that are managed on all systems.

Unity Alerts

Alerts need an appropriate response. Some alerts require immediate action for Unity to continue functioning. As part of Unity configuration and setup, configure critical monitoring notifications to see time-sensitive alerts immediately. Set up less-critical alerts for appropriate time intervals so they can be addressed before becoming critical.

Unity uses the Notification Service that is bundled with the Unity UI to send alerts using email or other monitoring paths for rapid response when needed.

Note:

This guide assumes that the Notification Service is configured. For assistance with that service, contact Teradata Customer Support.

Note:

For information about forwarding alerts to email, see *Teradata® Unity™ User Guide*.

If a non-critical alert appears regularly, contact Teradata Customer Support to determine if the alert can be disabled.

If a critical alert appears regularly, contact Teradata Customer Support to determine root cause.

Recommended Alerts

The following alerts are the minimum recommended alerts for email notification:

Alert Category	Alert Code	Severity	Suggested Response Time
Object interrupt	40033	Critical	Core hours
	40048	Critical	
	40053	Critical	
	40059	Critical	
	40020	Warning	Core hours
	40021	Critical	Immediate
System interrupt	40031	Critical	Immediate
	40047		
	40051		
	40064		

Alert Category	Alert Code	Severity	Suggested Response Time
Object unrecoverable	40032	Critical	Core hours
	40054		
System unrecoverable	40030	Critical	Immediate
	40052	Critical	Immediate
Recovery logon Failures	40034	Critical	Core hours
Query deadlock	40041	Critical	Immediate
Logon mismatch	40043	Critical	Immediate
No available systems	40044	Critical	Immediate
High availability	40103	Critical	Immediate
	40104		
	40102		
Network and process connectivity	40110	Critical	Immediate
	40160		
	40161		
	40168		
Blocked sessions (User defined)	40210	Warning	Core hours
	40215	Critical	Immediate
System resource	40201	Warning	Core hours
	40222		
	40223		
	40225		
	40226	Critical	Immediate
System access	40005	Critical	Immediate
	40035		
	40065		
	40100		

Setting Up Alerts

For information about setting up alerts, including forwarding alerts to email, see the *Teradata® Unity™ User Guide*

Checking Alerts

Checking Alerts in the Unity UI

1. In the **Unity** UI, select **Alerts**.
A list of alerts appears.
2. [Optional] Select the row for the alerts you want to expand, then select the the row of the alert to see alert details.

Checking Alerts Using unityadmin

1. At unityadmin, run the following command:

```
UNITYADMIN: ALERTS LIST;
```

2. Check for alerts that show Opened or Updated later than the listed health check, then do one of the following if needed:

Alert Status	Action
Large number of new alerts are not useful	a. Contact Teradata Customer Support for assistance.
Warning alerts you are not familiar with	a. Contact Teradata Customer Support for assistance.
New critical alerts that cannot be resolved	a. Open an incident.

Clearing Alerts

You can clear all alerts or individual alerts. Clearing an alert removes it from the alert list.

1. In the **Unity** UI, select **Alerts**.
2. Do one of the following:

Option	Description
Clear Alerts	a. On the row you want to clear, select <input type="checkbox"/> , then select Clear Alerts .
Clear Alert	a. Select an alert code. b. For the individual alert you want to clear, select <input type="checkbox"/> , then select Clear Alert .

3. [Optional] Enter the reason for clearing the alert.
4. Select **Clear**.

Manual Monitoring

Manually monitoring a Unity cluster on a regular basis shows you changes in status and reveals health trends. This information is valuable for performing maintenance or creating incidents.

Checking Unity System Health

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking system health.

1. Do one of the following:
 - Log on to the primary or standby Unity server as root user
 - Log on to the primary or standby Unity server as another user with sudo privileges.
2. At the Linux prompt, run the following command:

```
unity status
```

3. Confirm that the following processes are in a running status:
 - Watchdog
 - Sequencer
 - Endpoint
 - All Dispatcher processes
4. If a process is not running run the following command at the Linux prompt to start the process:

```
unity start <processName>
```

5. Do one of the following:

Status	Action
Process started	a. Investigate why the process was not running previously. b. If the reason the process was not running previously is unexpected or cannot be determined, contact Teradata Customer Support.
Process did not start	a. Do one of the following: <ul style="list-style-type: none"> • Check if the standby sequencer can reach the managed databases and primary sequencer. • Check if the primary sequencer can reach the managed databases and standby sequencer. b. Do one of the following:

Status	Action
	<ul style="list-style-type: none"> If a sequencer can reach the managed system and other sequencer, run the following command at the Linux prompt: <pre>unity start <processName></pre> If a sequencer cannot reach the managed databases and the sequencer, open an incident. <p>c. At the Linux prompt, run the following command to check how much disk space is left for the partition containing the /data/directory:</p> <pre>df -H</pre> <p>d. If the partition is more than 80% full, contact Teradata Customer Support for instructions on removing files in the /data/directory.</p> <p>e. At the Linux prompt, run the following command to check for TVC messages raised since the last check that may be of concern:</p> <pre>grep "Unity:" /var/log/messages</pre>

Checking DFS Space Usage

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking DFS space usage.

- To see how much space is left for the partition, do one of the following :

Unity Status	Action
Unity is running	<p>a. At unityadmin, run the following command:</p> <pre>DFS STATUS;</pre>
Unity is not running	<p>a. At the Linux prompt, run the following command:</p> <pre>df -H</pre>

- If the partition containing the /data/ directory is more than 80% full, recover halted or interrupted objects.
For assistance clearing files, contact Teradata Customer Support.

Checking HA Sync Status

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking HA Sync status.

Note:

If the Unity system mode is standalone, HA sync status does not apply.

1. At unityadmin, run the following command to check sequencer status:

```
STATUS;
```

2. Confirm the following expected values:

Status	Expected Value
HA Status	In Peer
(Primary Unity System):5344	ACTIVE
(Standby Unity System):5344	STANDBY (SYNCHRONIZED)
Active Count	1
Standby (Synchronized) Count	1

Checking Managed System Health

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking managed system health.

1. At unityadmin, run the following command:

```
STATUS;
```

The expected status for each system is running.

2. If the status is not running, do one of the following:

Status	Action
Out of Service or Read-Only	<ol style="list-style-type: none"> a. Confirm the system is not undergoing maintenance. b. If you need the system to process queries, contact an administrator, then run the following command from unityadmin if advised: <pre>SYSTEM RECOVER;</pre>
Unrecoverable	<ol style="list-style-type: none"> a. Do one of the following:

Status	Action
	<ul style="list-style-type: none"> • If the system was previously active, contact an administrator to confirm it was not deactivated. • If the system is a new system or unrecoverable because of user action, sync and activate the system. • If the system was not previously active and is not a new system, open an incident.
Interrupted (not previously interrupted)	<ol style="list-style-type: none"> a. Check alerts for root cause of the system interrupt. b. If the root cause is not apparent and does not resolve on its own, open an incident.
Disconnected	<ol style="list-style-type: none"> a. If there is no planned outage in progress, check connectivity between Unity and the disconnected system. See Responding to Critical Events. <p>Note: A system may become disconnected for any of the following reasons:</p> <ul style="list-style-type: none"> • Database for the managed system is not running. • Unity Management User is unable to log in. • Unity is unable to reach the managed system over the network. • Management system is not responding to status request from Unity because the system is busy or the database is unavailable.

Related Information:

[Basic Maintenance Concepts](#)

Checking Dispatcher HA

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking managed system health.

Note:

If the Unity system mode is standalone or dual standalone, do not check Dispatcher HA status.

1. At unityadmin, run the following command:

```
DISPATCHER STATUS;
```

2. Confirm the following dispatcher results:

- System Connectivity: Process is UP (active), Connected to a system
- System Connectivity: Process is UP (standby), Connected to a system

Checking Object Health

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking managed system health.

Note:

If you are using passive routing, you will not see objects in the active or any other state. This is expected.

1. At unityadmin, run the following command:

```
STATUS;
```

2. If the status for any system is not active, do the following:

Response or Result	Action
UE STATUS can only be executed on the active sequencer	<p>The sequencer is not the active sequencer.</p> <ol style="list-style-type: none"> a. Log on to the other sequencer in the cluster. b. At unityadmin, run one of the following commands: <ul style="list-style-type: none"> • View overall status of all components: <pre>STATUS;</pre> • See if the cluster is currently in peer state: <pre>HA STATUS;</pre>
Interrupted sessions	<ol style="list-style-type: none"> a. At unityadmin, run the following command: <pre>SESSION LIST INTERRUPTED;</pre> b. Make note of the error preventing recovery of the system and try to fix it. c. If you cannot fix the error, open an incident.
Interrupted objects	<ol style="list-style-type: none"> a. At unityadmin, run the following command: <pre>OBJECT LIST INTERRUPTED;</pre> b. Check the results against previous results. c. Try to fix the error causing interruption. See Interrupted Objects. d. Raise an incident in the following cases: <ul style="list-style-type: none"> • Objects have been interrupted during multiple recovery attempts and cannot be fixed. • The same object is interrupted on all Unity servers.
Unrecoverable objects	<ol style="list-style-type: none"> a. At unityadmin, run the following command: <pre>OBJECT LIST UNRECOVERABLE;</pre>

Response or Result	Action
	b. If there are new unrecoverable objects that were not deactivated by the DBA and you cannot identify the cause, open an incident.

Checking for New TVI Messages

Note:

If the Unity system is undergoing planned maintenance, wait until the maintenance window is complete before checking for new Teradata Vital Infrastructure (TVI) messages.

1. At the Linux prompt, run the following command:

```
grep "Unity:" /var/log/messages/
```

2. Review new messages since the last check.
3. If you need assistance with new messages, contact Teradata Customer Support.

Information from Log and Configuration Files

Unity generates log, message, and configuration files that provide valuable insight into Unity performance and help resolve issues.

Unity Log Files

Each Unity process maintains the following two log files in `/opt/teradata/unity/logs/`:

Log File Type	Description
Error	Contains all error, warning, and informational log messages for the process. The Active Sequencer Error log file contains error and warning level messages from all processes.
Debug	Contains the same information as the Error log file and trace-level log messages.

Unity rotates these log files automatically. For example, if the log file `example.log` reaches 100 MB, Unity moves the log information to a file named `example.log.0` and starts a new `example.log` file.

Unity continues rotating log files until they reach `example.log.9`, then archives those log files to `/var/opt/teradata/unity/oldLogs`.

Unity Diagnostic Files

When objects become interrupted, or when support files are generated, Unity generates diagnostic files for Teradata Customer Support.

Unity writes the diagnostic files to `/opt/teradata/unity/logs/`.

You can generate diagnostics by running the following command at `unityadmin`:

```
GET DIAGNOSTICS;
```

`/var/log/messages`

Unity writes critical alert messages to `/var/log/messages`.

You can view critical messages by running the following command at the Linux prompt:

```
grep "Unity:" /var/log/messages
```

Unity Configuration Files

Unity generates log, message, and configuration files that provide valuable insight into Unity performance and resolving issues.

Unity Properties File

The Unity properties file contains connection information that is required at startup and is a valuable resource for diagnosing startup issues. The properties file is located here: `/etc/opt/teradata/config/unity.properties`

See *Teradata® Unity™ Installation, Configuration, and Upgrade Guide for Customers*.

Unity Support Script

If Teradata Customer Support needs diagnostic information to determine the root cause of issues, you may be asked to run the Unity support script to create a bundle of logs and other data.

The support script is available at `/opt/teradata/unity/support/unity_support.sh`.

After starting the script, follow the script instructions to make note of file names that are part of the output.

Note:

For non-standalone Unity installations, support files are located on both Unity servers. You must have root-level privileges to run the script on the Unity servers in both regions.

Note:

When opening an incident with Teradata Customer Support, you must upload files from both Unity servers.

Unity Troubleshooting

Discovering common reasons for alerts and critical events helps you prevent more issues and keep your Unity system running more efficiently.

Interrupted Objects

An object becomes interrupted on a system if a SQL statement succeeds on at least one system but fails with an error on another system. This mismatch causes objects to become interrupted on the system with the error. Unity replays the SQL statement to attempt recovery. Some errors may recover automatically, although others require manual intervention.

If an object is interrupted on a system, and another SQL statement or session needs to use that object, any other objects used by that session or SQL statement also become interrupted. This is a cascade interrupt. Once the objects or sessions required for this object recover, the cascade interrupts recover as well.

The recovery may take a few attempts, but if no more root cause interrupts occur, the objects become active.

Finding Root Cause of Interrupted Objects

When an object becomes interrupted, the session also becomes interrupted. Looking at the alert Unity sends to the DBA shows you what the SQL statement was and any difference in response.

Common reasons for interrupted objects include the following:

- A GRANT statement was run on one system but not the other, causing a permissions violation on one system.
- One system running out of perm space for a query.
- One system running out of spool space for a query.
- An external load job failed and left a load lock on a table on one system.
- An external cleanup script dropped an object on one system.

1. At unityadmin, run the following command:

```
SESSION LIST INTERRUPTED;
```

Unity displays the session list. For example:

```
unityadmin> session list interrupted;
-----
Unity Session Number      : 1000
Unity Transaction ID      : 13
System                    : exampledb2
Root Cause                 : Yes
Interrupt Time            : 03/27 13:45:45
```

```

Time of Original Query   : 03/27 13:15:42
Error Code               : 3807
Error Text               : Object 'example' does not exist.
SQL Text                 : insert into example values (25);
Statement Type           : Insert

```

The error in this example indicates the object `example` does not exist on the system `exampledb2`. When you create the object `example` on the system, the interrupt recovers automatically.

Cascade Interrupts

If an object is interrupted on a system and another SQL statement or session needs to use that object, any other objects used by that session or SQL statement also become interrupted. This is a cascade interrupt.

You do not need to take any action to resolve cascade interrupts. Cascade interrupts resolve automatically when the object or session that caused the interrupt is fixed. Resolution may require multiple rounds of automatic recovery.

Example Cascade Interrupted Objects

A cascade alert indicates an object needs another object or session to recover before it can become active. For example:

```

R
Alert No.           : 27
Alert Type          : 40033
Alert Description   : A recovery operation on an object has been interrupted.
Recovery can continue when the current condition is resolved. Refer to the User
Guide for more information.
Alert Details       : Transaction 14 (session 1000) cannot be applied on system
1 (manageddb1) because session 1000 is interrupted. Table sample.cascade
is interrupted.

```

See [Finding Root Cause of Interrupted Objects](#).

Interrupted Object Alerts

Example Object Interrupt Alert

Every interrupted object raises an alert that remains open until the object is recovered and becomes active. For example:

```

Alert No.           : 21
Alert Type          : 40053
Alert Description   : An inconsistent response was detected from a Teradata
system. As a result the object has been placed in the Interrupted state.

```

Alert Details : System 1 (manageddb1) sent an inconsistent message, table sample.base is interrupted:

Transaction: 14

SQL: 'INSERT INTO base VALUES(1);'

Response 1 from system 1 (manageddb1): row count 0

Code: 3807

Message: Object 'base' does not exist.

Response 2 from system 2 (manageddb2): row count 1

Code: 0

Message: Success.

Note the alert text contains:

- The system it failed on (manageddb1)
- The object(s) it failed on (sample.base)
- The expected success message and code (success, 0)
- The actual failure message and code (Object 'base' does not exist, 3807)

The alert remains open until the SQL statement succeeds.

Example Object Interrupt Alert During Recovery Operation

Object interrupt alerts may be raised during recovery operations. For example:

Alert No. : 25

Alert Type : 40033

Alert Description : A recovery operation on an object has been interrupted. Recovery can continue when the current condition is resolved. Refer to the User Guide for more information.

Alert Details : System 1 (manageddb1) sent an inconsistent message, table: sample.base, session: 1000 are interrupted:

Transaction: 14

SQL: 'insert into base values(1);'

Expected response: [success] row count 1

Code: 0

Message: Success

Actual response: row count 0

Code: 3807

Message: Object 'base' does not exist.

Mass Interrupt Events

Due to cascade interrupts, a system may have several interrupted objects from a single root cause event. With automatic recovery enabled, the recommended default, Unity tries to recover all objects at set time intervals.

When a system or a large number of objects are interrupted, it may become a strain on Unity and managed system resources. Indicators of strain include the following:

- You fix the root cause but recovery is not progressing.
- Alert details show the root cause as being out of sessions or load slots.

To allow adequate time to analyze and correct any issues on the Teradata systems, temporarily disable automatic recovery. When the issues are resolved, restore automatic recovery.

Temporarily Disabling Automatic Recovery

When a system or a large number of objects are interrupted, it may become a strain on Unity and managed system resources. To allow adequate time to analyze and correct any issues on the Teradata systems, temporarily disable automatic recovery. When the issues are resolved, re-enable automatic recovery.

1. At unityadmin, run the following command:

```
CONFIG LIST KEY RECOVERYINTERVAL;
```

Unity returns the sequencer system information, including the default recovery interval of 300 seconds, and a custom recovery interval if applicable. For example:

```
Sequencer System Defaults
-----

RecoveryInterval: 300

Sequencer User Set Values
-----

RecoveryInterval: 600
```

In this example, Unity is using a custom recovery interval of 600 seconds.

When you initiate recovery, Unity sets the custom recovery interval back to the default.

2. If you are using a custom recovery interval, make note of the value so you can reset the value after recovery.
3. At unityadmin, run the following command to disable automatic recovery:

```
CONFIG UPDATE SEQUENCER RecoveryInterval 0 REASON 'Disabled Automatic
Recovery for incident _____';
```

4. At unityadmin, run the following command to confirm a recovery is not in progress:

```
SYSTEM RECOVERY STATUS;
```

5. At unityadmin, run the following command to make sure no objects on the system are being restored:

```
STATUS;
```

6. At unityadmin, run the following command to determine the cause of the mass interrupt event, then fix the errors:

```
SESSION LIST INTERRUPTED;
```

Note:

If you cannot fix the issues, contact Teradata Customer Support for assistance before proceeding with recovery.

7. At unityadmin, run the following command to start recovery for the specified system:

```
SYSTEM RECOVER <systemId>;
```

8. At unityadmin, run the following command periodically to check recovery status:

```
SYSTEM RECOVERY STATUS;
```

When recovery is complete, Unity returns the status `recovery not in progress`.

9. At unityadmin, run the following command after recovery is complete to check for new interrupts:

```
SESSION LIST INTERRUPTED;
```

10. Do one of the following:

Recovery Status	Action
No new interrupts	<p>a. At unityadmin, run one of the following commands to restore automatic recovery:</p> <ul style="list-style-type: none"> If you are using a custom recovery interval: <pre>CONFIG UPDATE SEQUENCER RECOVERYINTERVAL <recovery interval>;</pre> <p>This restores automatic recovery to the custom recovery interval specified.</p> If you are using the default recovery interval: <pre>CONFIG RESET SEQUENCER SEQUENCER RECOVERYINTERVAL;</pre> <p>This restores automatic recovery to the default recovery interval of 300 seconds..</p>
New errors	<p>a. Fix the issues, then repeat the recovery process.</p> <p>Note:</p> <p>If you cannot fix the issues, contact Teradata Customer Support for assistance before proceeding with another recovery attempt.</p>

Unrecoverable Objects

If a write statement succeeds on multiple managed systems and the result or row count also succeeds but with a different value, the following occurs:

- Data mismatch
- Query objects become unrecoverable on a managed system

Unity does not attempt recovery. The object must be manually synchronized to become active. Unity may also mark an object as unrecoverable if it detects a potentially unsafe operation.

Finding the Root Cause of Unrecoverable Objects

1. If an object has not been made intentionally unrecoverable, collect all DBQL for the object on all managed systems, then open an incident.

The following is an example of an unrecoverable object alert:

```
Alert No.           : 85
Alert Type          : 40032
Alert Description   : The object cannot be recovered and synchronized
                    : automatically by Teradata Unity. The object will need to be manually
                    : resynchronized. Refer to the User Guide for more information.
Alert Details       : System 1 (manageddb1) sent an inconsistent message,
                    : table sample.cascade is unrecoverable:
Transaction: 26
SQL: 'insert into cascade select * from base;'
Expected response:  [success] row count 1
Actual response:    [success] row count 0
```

Common Reasons for Interrupted Systems

Common reasons for interrupted systems include the following:

- Mismatch on certain DDL statements, for example CREATE DATABASE.

Repairing the mismatch will resolve this issue.

- Timeout while reading or writing to the system's heartbeat table.

For a heartbeat timeout, make sure that unitymgmtuser has no TASM restrictions and that no backup or restore jobs were running that could impact access to the Unity Management User.

- Timeout during automatic recovery of a long-running transaction.

If Unity does not receive a response within the specified recovery timeout, the system interrupts to allow the long-running transaction to complete, then restarts recovery. If this is an ongoing issue, contact Teradata Customer Support.

Critical Events

If you are experiencing critical events, review the recommended responses while Teradata Customer Support is preparing to engage.

If an error code does not indicate a specific problem with Unity, check for problems with the network or underlying databases.

Responding to Critical Events

1. Perform the following checks:

Check	Steps
Client-to-Unity connectivity	<ol style="list-style-type: none"> Do one of the following: <ul style="list-style-type: none"> If an application is not functioning, use a basic BTEQ connection to check the network. If possible, run BTEQ from a client system to Unity. If a BTEQ connection is not available, use a telnet connection to the Unity system with port 1025 to check for a listening port. If port 1025 is not available for a connection, contact your IT department. If a connection is not possible from the client system to Unity or is not working, use BTEQ to log on to the Unity system from the Unity system. <p>At the Linux prompt, run the following command with a managed database username and password to log in locally through Unity:</p> <pre>bteq .logon 127.0.0.1:1025/<user>;</pre> <p>Note: Specifying port 1025 is required when using BTEQ from the Unity system.</p> <p>If the login succeeds from the Unity system but fails from the client system, contact your IT department.</p>
Unity servers	<ol style="list-style-type: none"> Log on to each Unity server. At the Linux prompt, run the following command to confirm all processes are running: <pre>unity status</pre> Do one of the following: <ul style="list-style-type: none"> If the server cannot be reached, contact your IT department. If the server can be reached, confirm Unity processes are running. If they are not running, see Checking Unity System Health.
Unity operational status	<p>At least one managed system must be active for Unity to process requests.</p> <ol style="list-style-type: none"> At unityadmin, from the system you think is the primary, run the following command: <pre>STATUS;</pre> Do one of the following: <ul style="list-style-type: none"> If the system is not the primary system, run the command on the other Unity system to see if a failover has occurred. If neither system is primary, check network connectivity between Unity systems. If systems are disconnected, check managed system status and connectivity. See Checking Managed System Health.

Check	Steps
	<ul style="list-style-type: none"> If managed systems are interrupted, check alerts and sequencer logs. For more information about system interrupts, see Common Reasons for Interrupted Systems. If systems are unrecoverable, queries that use that managed system will fail. Investigate why a system is unrecoverable and open an incident if needed.
Unity-to-database connectivity	<p>a. To confirm connectivity is working between the Unity servers and the managed Teradata system, run the following command:</p> <pre>/opt/teradata/unity/scripts/connTest.sh -s <tdpid></pre> <p>You can repeat this command on each Unity server to confirm that server can connect.</p>
TVI alerts	<p>a. At the Linux prompt on the Unity primary server, run the following command to check for critical alert messages:</p> <pre>grep "Unity:" /var/log/messages</pre>
Critical alerts	<p>At unityadmin, run the following command to check for critical alerts:</p> <pre>ALERT LIST;</pre>