

# Teradata Vantage™ - Application Programming Reference

---

Release 17.10

July 2021

# Copyright and Trademarks

Copyright © 2000 - 2021 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](mailto: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

<b>Chapter 1: Introduction to Teradata Vantage Application Programming</b>	<b>5</b>
Related Documentation	5
Changes and Additions	5
<b>Chapter 2: Concepts</b>	<b>6</b>
Workload Management API	6
Automated Statistics Management API	10
<b>Chapter 3: Workload Management: PM/API</b>	<b>13</b>
PM/API Processing	13
Comparisons Between PM/API and Teradata SQL Requests	18
PM/API Dynamic Data	20
<b>Chapter 4: Workload Management API Features and Examples</b>	<b>24</b>
System PMPC API Features	24
Teradata Dynamic Workload Management API Features	31
Query Band API Features	34
Embedded Services System API Features	36
<b>Chapter 5: System PMPC APIs</b>	<b>38</b>
PM/APIs	38
Open APIs (SQL Interfaces)	188
<b>Chapter 6: Teradata Dynamic Workload Management APIs: PM/APIs</b>	<b>301</b>
PM/APIs	301
Open APIs (SQL Interfaces)	357
<b>Chapter 7: Workload Management: Query Band APIs</b>	<b>404</b>
PM/API	404
Open APIs (SQL Interfaces)	407
<b>Chapter 8: Workload Management: Embedded Services System APIs</b>	<b>420</b>
Open APIs (SQL Interfaces)	420
<b>Chapter 9: Workload Management: Ruleset APIs</b>	<b>427</b>
TDWMCreateSystemThrottle	427
TDWMCreateArrivalRateMeter	429
TDWMAddClassificationForRule	430
TDWMAddClassificationForTarget	435

TDWMAddLimitForRuleState .....	438
TDWMMManageRule .....	439
TDWMActivateRuleset .....	440
TDWMDeleteRule .....	441
Examples .....	441
<b>Chapter 10: Automated Statistics Management API Features and Examples .....</b>	<b>445</b>
Automating Statistics Collection and Monitoring .....	445
Analyzing User Objects and Logged Query Plans .....	447
Preparing and Collecting Statistics .....	448
Controlling Recommended Actions and Overriding Default Settings .....	449
Creating Object Lists .....	457
Creating and Maintaining Query Lists .....	458
<b>Chapter 11: Automated Statistics Management APIs .....</b>	<b>459</b>
Granting Required Privileges .....	459
GrantPrivileges .....	460
Automate Open APIs .....	461
Analyzer Open APIs .....	485
Collect Open APIs .....	512
DBAControl Open APIs for Excluding Objects .....	533
DBAControl Open APIs for Statistic Settings .....	537
DBAControl Open APIs for Prepared Collections .....	543
DBAControl Open APIs for Managing Space .....	552
Open APIs for Object Lists .....	558
Open APIs for Query Lists .....	564
<b>Appendix A: Sample PM/API Application .....</b>	<b>570</b>
<b>Appendix B: MONITOR SESSION Response Combinations .....</b>	<b>600</b>
<b>Appendix C: Additional Information .....</b>	<b>603</b>

# Introduction to Teradata Vantage Application Programming

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.

This is a reference document for the Teradata Vantage application programming interfaces (APIs). It describes the following types of APIs and how they can be used to interface with your own custom applications when other tools (such as, resource usage reports) or client applications do not provide the information you need:

- [Workload Management API](#)

Workload management APIs can be used for creating third-party applications and custom applications requiring the acquisition and use of Teradata workload management and performance data.

Note that the workload management PM/APIs described in this document use monitor software versions 11 and earlier (see [MONITOR VERSION](#) for details). These PM/APIs allow you to create new applications and enhance existing custom applications.

- [Automated Statistics Management API](#)

The Automated Statistics Management open APIs can be issued from the Teradata Viewpoint Stats Manager portlet or directly from within customer applications. They can be used to accurately identify unused or out of date statistics, cardinality estimation errors, and override SQL COLLECT STATISTICS statement settings in certain well-defined scenarios.

## Related Documentation

- If your custom application uses an earlier monitor version, depending on the version used, see previous releases of this document: *Workload Management API: PM/API and Open API* or *Application Programming Reference: Workload Management*.
- For information on Teradata Viewpoint Stats Manager portlet, see *Teradata Viewpoint User Guide*.
- For more information about the workload management (WM) capacity on demand (COD) feature, see *Workload Management Capacity on Demand and Other Hard Limits Orange Book*, TDN0009761.

## Changes and Additions

Date	Description
July 2021	<ul style="list-style-type: none"> <li>• Updated the descriptions of IOCompleted and IOCompletedKB.</li> <li>• Added information on new Workload Management ruleset APIs. See <a href="#">Workload Management: Ruleset APIs</a>.</li> <li>• Added information on the Arrival Rate Meter.</li> </ul>

# Concepts

The following describes the different types of APIs and the requirements for using them.

## Workload Management API

Workload management API consists of interfaces to PM/APIs and open APIs. You can use these interfaces to:

- Monitor system and session-level activities.
- Monitor Teradata Active System Management (ASM) activity.
- Track system usage and manage task priorities.
- Modify Workload Management ruleset.

## API Categories

PM/APIs and open APIs, also called SQL interfaces, are divided into the following categories:

- System PMPC
- Teradata Dynamic Workload Management
- Query Band
- Ruleset

Embedded services system is another category that applies only to some of the SQL interfaces (for example, the GetPSFVersion and TD\_get\_COD\_limits functions).

The following diagram shows how:

- The System PMPC, Teradata Dynamic Workload Management, and Query Band PM/API requests (such as Call-Level Interface Version 2 [CLIV2] and the Teradata JDBC Driver) interface to the PMPC subsystem through the MONITOR partition.

---

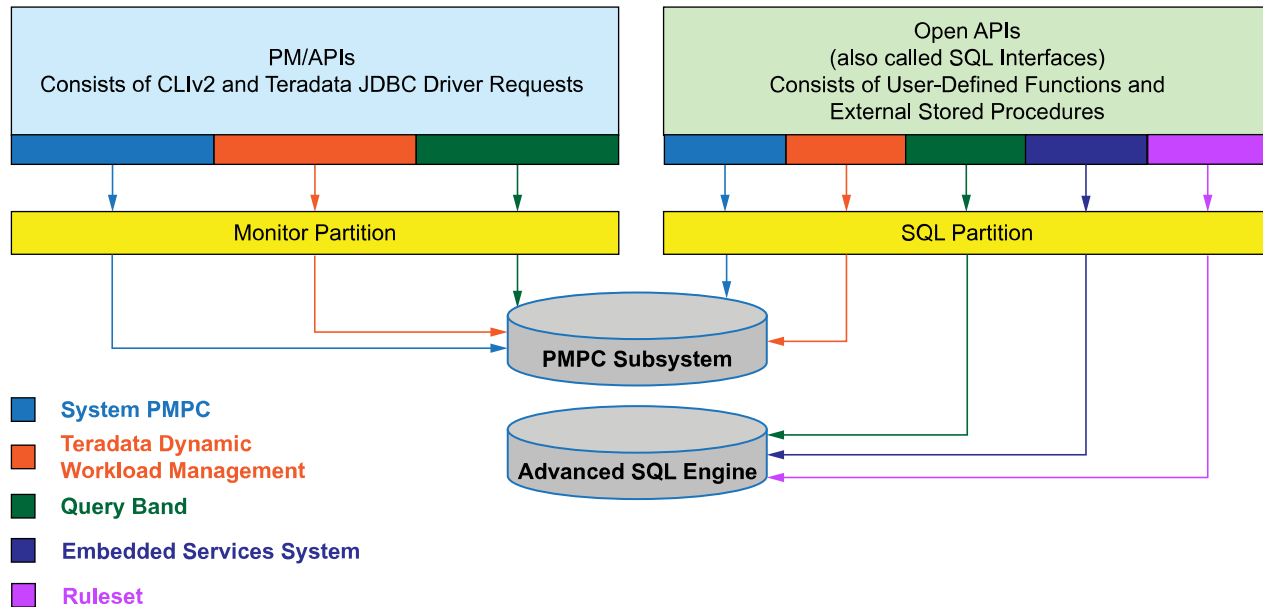
### Note:

These PM/APIs are supported by both Call-Level Interface Version 2 (CLIV2) and the Teradata JDBC Driver. For more information on how the Teradata JDBC Driver requests are executed, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

---

- The System PMPC and Teradata Dynamic Workload Management SQL interfaces (such as, user-defined functions and external stored procedures) connect to the PMPC subsystem through the SQL partition, except TDWMApply, TDWMRuleControl, and TDWMSetLimits. These APIs connect to the database system through the SQL partition and are not shown in the diagram below.

- The embedded services system and most Query Band SQL interfaces connect to the SQL partition, except MonitorQueryband. This SQL interface is the only Query Band API that connects to the PMPC subsystem through the SQL partition and is not shown in the diagram below.



For more information on these APIs, see:

- [Workload Management API Features and Examples](#)
- [System PMPC APIs](#)
- [Teradata Dynamic Workload Management APIs: PM/APIs](#)
- [Workload Management: Query Band APIs](#)
- [Workload Management: Embedded Services System APIs](#)
- [Workload Management: Ruleset APIs](#)

## PM/APIs

PM/APIs provide access to PMPC routines resident in Vantage. The PMPC subsystem is available through a logon partition called MONITOR, using a specialized PM/ API subset of CLIV2 or Teradata JDBC Driver.

PM/APIs have the following features:

- CLIV2 or Teradata JDBC Driver data is acquired in near real time, with less overhead and minimal possibility of being blocked. These capabilities allow frequent in-process performance analysis.
- CLIV2 request saves the raw data in an in-memory buffer where a client application program can easily retrieve the data for real-time analysis or importing into custom reports. The Teradata JDBC Driver returns the data as a JDBC ResultSet where a client application program can easily retrieve the data.
- CLIV2 or Teradata JDBC Driver request provides access to data that the resource usage does not. For example, session-level resource usage data, and data on application locks and which application is being blocked.

Using PM/APIs may not be the right choice for all performance monitoring requirements. Standard performance monitoring tools and reports, such as resource usage reports, may be sufficient.

## Open APIs

The workload management open API provides an SQL interface to the PMPC subsystem and Teradata system through user-defined functions, embedded services functions, and external stored procedures. Most of the SQL interfaces available to the PMPC subsystem provide similar functionality to the CLIV2 or Teradata JDBC Driver requests.

### Note:

Most open APIs do not follow transaction rules. If a transaction calls a UDF or external stored procedure and the transaction rolls back, the action of the UDF or external stored procedure is not rolled back. However, the external stored procedures that update the TDWM database must follow the transaction rules. If a transaction calls one of these external stored procedures and the transaction is aborted, the update will be rolled back.

## Differences Between Open APIs and PM/APIs

The following table describes the differences between open APIs (that is, SQL interfaces consisting of UDFs or external stored procedures) and PM/APIs.

Open APIs...	PM/APIs...
are issued in the current SQL partition	require logging on to the MONITOR partition. To view a diagram of the process, see <a href="#">API Categories</a> .
require EXECUTE privilege on the function or external stored procedure	require MONITOR privileges.
use SQL parsing, dispatching steps, and UDF processing	require use of a custom application in C, Java, or some other programming language.
run in priority of account string or by the Teradata dynamic workload management software classification	run in the system priority.
can be placed on a Teradata dynamic workload management software delay queue and can block system resources	do not block.
use AMP Worker Tasks (AWTs)	are not subject to running out of resources (AWTs).

## PM/APIs Requirements for Using the API

If you are using CLIV2, you must install CLIV2 on a client machine where the PM/API application is running.

If you are using the Teradata JDBC Driver, you must import the SQL Interface package and load the Teradata JDBC Driver. For complete instructions, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.



## Open APIs (SQL Interfaces) Requirements for Using the API

The SQL interfaces to workload management already exist if the following scripts have been run as part of Teradata installation:

- For System PMPC and Teradata dynamic workload management SQL interfaces, you must run Database Initialization Program (DIP) utility and execute the DIPDEM and DIPTDWM scripts.
- For the embedded services system functions, you must run the DIP utility and execute the DIPALL or DIPSYSFNC script. These scripts create the TD\_SYSFNLIB database which should only be used by the system to support the embedded services functions. For more information about activating the embedded services system functions, see *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

For information about these DIP scripts, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## Required Privileges

Each API described in this document, except for the MONITOR VERSION request and the embedded services system functions, has its own required privileges.

Below are some examples of the required privileges.

- To connect with the Teradata JDBC Driver, it is necessary that you are granted privileges for executing PM/API requests. For example, you must issue GRANT MONITOR TO guest before connecting as a guest user and executing any PM/API request.
- To issue the ABORT SESSION and MONITOR SESSION requests, you must have the ABORTSESSION and MONSESSION privileges respectively as part of your default role or these privileges must be granted directly to you.
- To issue the MONITOR AWT RESOURCE, MONITOR VIRTUAL RESOURCE, and MONITOR PHYSICAL RESOURCE requests, you must have the MONRESOURCE privilege as part of your default role or this privilege must be granted directly to you.
- To access the UDFs and external stored procedures, the DBA must grant EXECUTE FUNCTION and EXECUTE PROCEDURE privileges to you. These privileges are not granted by default.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Related Information

For more information on ...	See ...
the different System PMPC PM/APIs and open APIs	<a href="#">System PMPC APIs.</a>

For more information on ...	See ...
the different Teradata Dynamic Workload Management PM/APIs and open APIs	<a href="#">Teradata Dynamic Workload Management APIs: PM/APIs.</a>
the different Workload Management ruleset APIs	<a href="#">Workload Management: Ruleset APIs</a>
the different Query band PM/APIs and open APIs	<a href="#">Workload Management: Query Band APIs.</a>
the different embedded services system functions	<a href="#">Workload Management: Embedded Services System APIs.</a> For more information about embedded services system functions, see <i>Teradata Vantage™ - SQL Functions, Expressions, and Predicates</i> , B035-1145.
how to code an application that uses the CLlv2 requests in this document	<ul style="list-style-type: none"> <li>• <i>Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems</i>, B035-2417.</li> <li>• <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418.</li> </ul>
using the Teradata JDBC Driver to access the database	<i>Teradata JDBC Driver Reference</i> , available at <a href="https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html">https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html</a> .

## Automated Statistics Management API

Automated Statistics Management API consist of interfaces to open APIs. You can use these open APIs to:

- Identify situation where statistics management might be improved (for example, missing, out of date (also referred to as stale), and unused statistics).
- Ability to control the application of recommended actions by the Analyzer operation and override any of the default settings stored in TDSTATS database.
- View and modify the prepared collection list prepared by the Automated Statistics Management PrepCollect open API prior to its actual submission.

### Note:

Teradata recommends that you interface with these APIs directly through Teradata Viewpoint Stats Manager portlet. If you require a specific, customized API that the Teradata Viewpoint Stats Manager portlet does not provide, you can call the Automated Statistics Management APIs directly from your own application.

### API Categories

Automate Statistic Management open APIs are divided into the following categories:

Category	Description
Automate	These open APIs copy the qualifying statistics definitions from the DBC dictionary to the TDSTATS database and synchronize any subsequent changes from SQL Data Definition Language operations. For more information, see <a href="#">Automating Statistics Collection and Monitoring</a> .
Analyzer	These open APIs analyze user objects and logged query plans to identify missing, stale, and unused statistics. For more information, see <a href="#">Analyzing User Objects and Logged Query Plans</a> .
Collect	These open APIs: <ul style="list-style-type: none"> <li>• Batch and prioritize a list of SQL COLLECT STATISTICS statements.</li> <li>• Submit the queued list of SQL COLLECT STATISTICS statements prepared by the PrepCollect open API.</li> </ul> For more information, see <a href="#">Preparing and Collecting Statistics</a> .
DBAControl	These open APIs: <ul style="list-style-type: none"> <li>• Control the application of recommended actions by Analyzer open APIs.</li> <li>• Override any of the default settings stored in the TDSTATS database.</li> </ul> For more information, see <a href="#">Controlling Recommended Actions and Overriding Default Settings</a> .

Each category of open API records its results within the TDSTATS database which can be queried using SQL.

## Specifying the Scope of API Operations

All of the above listed categories of APIs accept input parameters that optionally limit the scope of the operation to a specified database, table, or statistic. You can also define a named list consisting of any set of databases or tables, and then call the APIs with the list name, see [Creating Object Lists](#) or [Open APIs for Object Lists](#).

## Requirements for Using the Open API

The SQL interfaces to statistics management already exist if the TDSTATS database has been created by the DIPSTATS script as part of the database installation.

You can use Teradata Studio or Teradata Studio Express to list the TDSTATS tables and views and view details about each view and table column. For information about Teradata tools and to download the tools, go to [Teradata Downloads](#).

## Required Privileges for Using the Open APIs

You must have the following privilege as part of your default role or it must be granted directly to you...	To ...
GRANT EXECUTE PROCEDURE ON TDSTATS TO User	call the Automated Statistic Management open APIs.

You must have the following privilege as part of your default role or it must be granted directly to you...	To ...
GRANT SELECT ON TDSTATS TO <i>User</i>	query the TDSTATS database directly. <b>Note:</b> If the Automated Statistics Management report open APIs (such as, AutomateReport, AnalyzeStatsReport, AnalyzeStatsUsageReport, and so on) are sufficient, you do not need to grant the SELECT privileges on the TDSTATS database.
GRANT EXECUTE ON DBC.dbqlaccessmacro TO <i>User</i>	enable statistics usage logging.
GRANT STATISTICS ON <i>User_Objects</i> TO TDSTATS	execute the AnalyzeStats or AnalyzeStatsUsage and RunCollect open APIs.

## Related Information

For information about ...	See ...
the functionality of these open APIs	<a href="#">Automated Statistics Management API Features and Examples.</a>
the required privileges to run the Automated Statistics Management open APIs	<a href="#">GrantPrivileges.</a>
each of the Automated Statistics Management open APIs	<a href="#">Automated Statistics Management APIs.</a>
Teradata Viewpoint	<i>Teradata® Viewpoint User Guide</i> , B035-2206.

# Workload Management: PM/API

The following describes how each MONITOR request processes the following PM/APIs issued to Vantage by an end-user application interface on a client:

- System PMPC
- Teradata Dynamic Workload Management
- Query Band

The client can be either a mainframe-attached or workstation-attached.

## PM/API Processing

You can use the following Teradata proprietary APIs and libraries to write end-user application interfaces to PM/API routines.

- CLlv2, an interface between the end-user application written in C and the Teradata Director Program. Teradata Director Program is the interface between CLlv2 and the database. For an example of CLlv2, see [Sample PM/API Application](#).
- The Teradata JDBC Driver, an interface between the end-user application written in Java and the database.

Whether the client application interface you develop is a simple utility or a complex system, the process is the same. Because the PM/API is accessed by either CLlv2 or the Teradata JDBC Driver, requests and responses are processed in much the same way as for a Teradata SQL application. No changes are required in CLlv2 or the Teradata JDBC Driver to support PM/API. As a result, PM/API capabilities are available on any client platform that supports CLlv2 or the Teradata JDBC Driver.

## Related Information

For details on the uses of ...	See ...
CLlv2	<ul style="list-style-type: none"> <li>• <i>Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems</i>, B035-2417</li> <li>• <i>Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems</i>, B035-2418</li> </ul>
the Teradata JDBC Driver	<i>Teradata JDBC Driver Reference</i> , available at <a href="https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html">https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html</a>

## Creating a Request with CLlv2

Your client application program tells CLlv2 what to do by creating a request string consisting of:

- A request parcel (when the response is desired in record mode) or an IndicReq parcel (when the response is desired in indicator mode).

When the response mode option is set in CLI or CLlv2, it automatically uses the correct request parcel format.

- A USING Data String, which contains the input data.

Unlike Teradata SQL, PM/API does not use a USING Phrase to name the variables and reserve space in the request parcel. Instead, each MONITOR request has a USING Data String of a particular fixed format that determines the order of items, their data types, and lengths.

Because a USING Data String is required, either a data parcel must follow a request parcel or an IndicData parcel must follow an IndicReq parcel.

An IndicData parcel is recommended, because several of the fields in the USING Data String can be NULL.

Generating an IndicReq parcel results in a response that contains a PclDataInfo parcel, which describes the number of response columns. Each Record parcel returned begins with a number of presence bits, that supply the NULL indicators for the result columns.

To pass PM/API requests to the database as the text portion (body field) of the request parcel, the application program calls the CLlv2 DBCHCL routine with the DBCAREA function code (4) set to the Initiate Request operation.

Code your application program to do the following before calling CLlv2 for the Initiate Request operation:

- Set the request pointer to the address of a character string containing the request name.

---

**Note:**

For the IDENTIFY request, set the request pointer to the address of a character string containing one of the following:

- IDENTIFY SESSION
  - IDENTIFY DATABASE
  - IDENTIFY USER
  - IDENTIFY TABLE
- 

- Set the request length to the length in bytes of the character string.
- Set the USING Data pointer to the address of the USING Data String.
- Set the USING Data Length to the length in bytes of the USING Data String.

### Setting Indicator or Record Mode

You can send the USING Data String in either indicator mode or record mode. CLI must inform CLlv2 which mode is used by setting the appropriate use presence bits option in the DBCAREA.

**Note:**

The use presence bits option is usually set to a default value for the site.

If you want...	Do the following ...
to change the default value	<ol style="list-style-type: none"> <li>1. Set change options to Y in the application program.</li> <li>2. Set the use presence bits option in the DBCAREA to Y.</li> </ol> <p><b>Note:</b> This also allows the application program to send NULL data to the Teradata system.</p>
the USING Data String sent in record mode	<ol style="list-style-type: none"> <li>1. Set change options to Y in the application program.</li> <li>2. Set the use presence bits option to N in the DBCAREA in the application program.</li> </ol>

## Executing a Request with the Teradata JDBC Driver

The client application uses a PreparedStatement, an object that represents a prepared SQL statement, for the PM/API requests. The client application:

- Specifies the PM/API request text as the Connection PreparedStatement method's SQL argument. The Connection interface represents a session with a specific database.
- Is not permitted to specify multiple requests separated by semicolons.
- Can only specify a single PM/API request.
- Specifies the input data of the PM/API request with the PreparedStatement setter methods (for example, setString or setObject).
- Must use the PreparedStatement execute() method to execute the PM/API request because each PM/API request may return multiple results. The execute() method handles these complex results.

For more information on the PreparedStatement and Connection methods, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Response Processing with CLlv2

PM/API requests are processed similarly to Teradata SQL requests because CLlv2 is used for both. Response processing is similar in both parcel ordering and buffer allocation. For an example CLI program, see [Sample PM/API Application](#).

Just as input data can be sent to the database in indicator or record mode, the response can be returned from the database in either indicator or record mode, depending on how the response mode option is set in CLlv2. You can set response mode to be independent of input data mode. For example, it may be more convenient to send the input data in record mode, but the response may require indicator mode if NULL is expected.

Response mode is usually set to a default value for the site. If you want to change the default value set, change options to Y in the application program.

If you want to return data in...	Set the response mode option in DBCAREA to...
indicator mode	I for <b>Indicator</b> . CLlv2 automatically uses the correct request parcel format.
record mode	R for <b>Record</b> . CLlv2 automatically uses the correct request parcel format.

**Note:**

Field mode, represented by F, is not supported in a PM/API request.

## Retrieving Results with the Teradata JDBC Driver

Each PM/API request may return multiple results. The client application must use the PreparedStatement execute() method to execute the PM/API request.

The client application calls the PreparedStatement getResultSet() method to retrieve the ResultSet from the PM/API request. A ResultSet provides access to a table of data generated by executing a PM/API request. The table rows are retrieved in sequence. Within a row its column values can be accessed in any order.

If the PM/API request returned multiple results, the client application calls the PreparedStatement getMoreResults() method to advance to the next result. Then, the client application calls the PreparedStatement getResultSet() method to retrieve the result set.

For more information on the PreparedStatement methods, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Parcel Ordering

The response returned from a MONITOR request is a series of parcels constructed by the database and sent to the client.

The basic parcel layout for a successful response is the same for most MONITOR requests, except the following:

- MONITOR SQL
- MONITOR VERSION

Some MONITOR requests such as:

- MONITOR SESSION
- MONITOR VIRTUAL CONFIG
- MONITOR PHYSICAL CONFIG
- MONITOR VIRTUAL RESOURCE
- MONITOR PHYSICAL RESOURCE

behave like multistatement requests, for which multiple record parcels are returned. For request-specific parcel layout, see [System PMPC APIs](#).



The following table shows the parcel layout for an unsuccessful response is the same for all MONITOR requests -- either a Failure or an Error parcel is returned.

Parcel Name	Parcel Flavor	Field Length	Comments/Key Parcel Body Fields
Failure	9	15 to 267	Message code and message text: description of cause of failure; request could not be processed. For example, the request is aborted.
Error	49	15 to 267	Message code and message text: description of cause of error. Application can fix the problem and resubmit the request. For example, response buffer is too small to hold entire response row.

## Buffer Allocation with CLIV2

A PM/API application, like a Teradata SQL application, requires space for a:

- Response buffer, containing the parcels transmitted back from the database.
- Request buffer, containing the parcels sent to the database.

Space for these two buffers is allocated for the application by CLIV2 based on the setting of DBCAREA arguments at the time a session is established. Default buffer sizes are controlled by the HSHSPB (control block containing site specific information), which is created during installation when CLI defaults are specified. The minimum size of a response buffer is 32,000 bytes.

If your response buffer is not large enough, your application program may receive an error message. For details on error messages, see *Teradata Vantage™ - Database Messages*, B035-1096.

## Logging On

A PM/API application logs on similar to a Teradata SQL application. However, there are some differences.

The partition name is MONITOR:

- In a Java application, specify the Teradata JDBC Driver PARTITION=MONITOR connection parameter. For details, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.
- In a CLIV2 application, set the following:
  - Run pointer to the address of the character string containing the word MONITOR.
  - Run length to the length in bytes of the character string containing the word MONITOR.

Your logon fails if:

- You do not have the appropriate MONITOR privilege.
- The PE or client that the MONITOR session is logged on to is already supporting its maximum number of four MONITOR sessions.

**Note:**

Currently on the PE, the load balancing mechanism does not support MONITOR partition sessions. Session Control allocates MONITOR session requests to the PE on which a MONITOR session is logged until the four sessions per PE limit is reached.

- The system-wide limitation of 128 concurrent MONITOR sessions is reached.

**Logging Off**

A PM/API application logs off the same way a Teradata SQL application does with one exception. If you log off a session that has a local session monitoring rate (SesMonitorLoc) of nonzero, the rate is changed to zero. This change is added to the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view), which is similar to what occurs if you had issued a SET SESSION RATE request prior to logging off.

A Java application calls the Connection close() method of the Teradata JDBC Driver to log off. The Connection close() method immediately releases a database and the JDBC resources instead of waiting for them to be automatically released. For more information on the Connection close() method, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

**Warning and Error Messages**

For a general discussion on common warning and error messages that may be returned in response to various requests, see [Common Warning and Error Messages](#).

For detailed information on the meaning of individual error and warning messages and the response required, see *Teradata Vantage™ - Database Messages*, B035-1096.

## Comparisons Between PM/API and Teradata SQL Requests

There are both similarities and differences between PM/API applications and Teradata SQL applications, such as Basic Teradata Query (BTEQ), special utilities (for example, Teradata FastLoad), and third-party software.

**Similarities**

The similarities between a PM/API based application and a Teradata SQL-based application include:

- Data types supported for input and output data in PM/API applications are a subset of those data types supported by Teradata SQL.
- Fixed length character data is blank padded on the right.
- NULL returned for the PclRecordType response parcel is equal to NULL returned by Teradata SQL (see [Response Groups](#)).

## Differences

The differences between a PM/API based application and a Teradata SQL-based application are:

- A PM/API application issues requests through a session partition named MONITOR.
- The MONITOR partition does not support the capabilities found in a Teradata SQL partition. The following table shows how the MONITOR partition reacts to the Teradata SQL partition capabilities.

If you use the Teradata SQL partition capability...	The MONITOR partition returns ...
Keep Response processing mode	an error message.
Execute a rewind operation	
Run the start-up operation	
Field mode request	
CleanUp request	
P (Prepare) or S (Setup) request processing option of the DBCAREA	

- Keywords used by the MONITOR PM/API requests are different from keywords in Teradata SQL. However, both have the same rules for identifiers.
- MONITOR query processing, unlike Teradata SQL, produces dynamic data.
- When using CLIV2, a Using Data String, which defines the input data, is always required in a PM/API application. In Teradata SQL, it is optional.

## Keywords/Reserved Words

The following keywords are database-reserved words:

- ABORTSESSION
- MONITOR
- MONRESOURCE
- MONSESSION
- SETRESRATE
- SETSESSRATE

The following keywords are database non-reserved words that are permitted as object names, but discouraged because of the possible confusion that may result:

- MONSQL
- MONVERSION

For a complete list of words that are unavailable for use as object names, see *Teradata Vantage™ - SQL Fundamentals*, B035-1141.

## Using Teradata SQL GRANT and REVOKE Statements

If you are using a Teradata SQL-based application, you can issue the GRANT and REVOKE statements. For more information on these SQL statements, see *Teradata Vantage™ - SQL Data Control Language*, B035-1149.

In rare cases, a change in privileges could temporarily block the request of an active MONITOR session because the system must look up the new privileges before accepting any more requests from the affected user. If the system query is blocked by a lock on a database dictionary table or if there is some other type of system processing bottleneck, you cannot issue any MONITOR requests until you acquire the new set of privileges. However, MONITOR requests submitted from the system console are not blocked by a change in privileges, because no privileges are necessary to issue MONITOR requests on the system console.

## Using the ABORT SESSION Request

If you are using a PM/API based application, you can issue the CLv2 or the Teradata JDBC Driver ABORT SESSION request.

With the proper privileges granted, you can abort and log off a database user.

If the ABORT SESSION command...	The database user receives...
targets a current database request or session	an error message that the transaction is aborted.
targets a request or session after the user has logged off from the database	an error message that the session is not currently logged on.

The error message is not returned to the user of the affected session until ABORT SESSION completes rolling back the current transaction or, if applicable, the session is logged off.

The status of an active session determines when the error message is received.

If, at the time you issue an ABORT REQUEST statement and...	The...
a request is outstanding for the user session	error message is sent as a response to the current request.
no request is outstanding, but a session is in progress	user receives the error message when the next request is received.

For more information on this request, see [ABORT SESSION](#).

## PM/API Dynamic Data

When you issue a request through the Monitor partition, current and dynamic data is reported. PM/API requests place data in a global repository (in memory), not in an intermediary spool file (on disk). AMP, PE, node, and session-level usage data are each collected in independent data collection global areas.

As a result, changes in the resource collection rate have no impact on session-level data, and vice versa. Any MONITOR request, except MONITOR VIRTUAL CONFIG, MONITOR PHYSICAL CONFIG, and IDENTIFY, may cause the memory repository to be updated on demand (specifically, once each collection period). All users share the data in the repository, which is used to generate responses to both queries and CONTINUE requests.

---

**Note:**

If, after issuing a PM/API request, the Monitor partition session returns an error parcel with empty error text, run the Database Initialization Program (DIP) option 1 (DIPERR), wait up to 60 minutes to reload the error text cache in the Monitor partition. After the error text cache is reloaded, the error parcels will have the proper error texts. For instructions on how to run DIP, see *Teradata Vantage™ - Database Utilities*, B035-1102.

---

## Monitoring Rates

You should be aware of the rate at which your PM/API client application requests performance data. If a PM/API client application requests data more frequently than the corresponding data collection rate, an individual request could potentially include data from one collection period mixed in with data from a subsequent collection period initiated by a different MONITOR user because common global data is shared.

However, the impact need not be detrimental. The data collected from one request when compared with the next request may not show a change in resource usage. Follow the simple rule that the PM/API client application must request data at the same rate, or a less frequent rate, than the rate at which the database collects that data.

The monitoring rate is a PM/API collection rate that sets the interval in seconds at which resource usage data is collected within memory.

---

**Note:**

A physical monitoring rate is the same as a virtual monitoring rate.

---

There are significant differences in the way resource usage and session utilization data are collected and reported by the MONITOR partition. For details, see [Data Collection](#) and [Collection and Logging Rates](#).

## Session-Level Data

Session-level data is collected cumulatively. The collection period is used to limit the frequency of this update.

For example, if you set the SET SESSION RATE to 120 seconds and issue a MONITOR SESSION request every 120 seconds, session usage data is collected and cumulatively totaled every 120 seconds.

---

**Note:**

Session-level data is lost in the event of a system outage.

---

Data reported includes data for the beginning 120 seconds as well as for subsequent intervals.

For more information on how session usage data is collected, see [System-Level Monitoring](#).

## Resource Usage Data

Resource usage data is collected based on activity during a collection period and does not reflect cumulative data for a sequence of collection periods. For example, if you set the SET RESOURCE RATE to 120 seconds and issue a MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request, resource usage data is collected at 120-second intervals.

If you do not examine the data within 120 seconds or enable resource usage tables for logging, it is lost when overwritten by data collected during the next 120-second collection interval.

## Features of Using Resource Usage Data

Resource usage data features are:

- Access is via SQL, not C or some other programming language
- More detailed
- Written to tables so past data values are available
- Can be accumulated over a long period of time and can be used for the following:
  - Examining trends and patterns
  - Planning system upgrades
  - Deciding when to add new applications to heavily utilized systems
  - Building baseline resource usage profiles for operations

## Logging Resource Usage Data

You can retain collected data for subsequent historical analysis by enabling one or more resource usage tables for logging. The table and type of PM/API request are listed below. For details on the resource usage tables and pre-defined report macros, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

To control the resource usage logging option (for example, setting the logging rate and enabling one or more log tables or subtables), use the ctl utility. For instructions on using ctl, see *Teradata Vantage™ - Database Utilities*, B035-1102.

PM/API Request	Table
MONITOR PHYSICAL RESOURCE MONITOR PHYSICAL SUMMARY	ResUsageSpma
MONITOR PHYSICAL RESOURCE MONITOR VIRTUAL RESOURCE	ResUsageShst
MONITOR VIRTUAL RESOURCE MONITOR VIRTUAL SUMMARY	ResUsageSldv (Storage Devices)
EVENT STATUS MONITOR WD	ResUsageSps

PM/API Request	Table
MONITOR VIRTUAL RESOURCE MONITOR VIRTUAL SUMMARY	ResUsageSvpr
MONITOR VIRTUAL RESOURCE	ResUsageSvdsd

## Collection and Logging Rates

You can control the rate at which resources are monitored (collected) and, if you enable resource logging, the interval at which a data row is to be inserted into the log table.

### Setting Collection Rates

The following collection rates can be set:

- Global session monitoring
- Local session monitoring
- Resource monitoring

For more information on these types of rates, see [Data Collection](#).

### Setting Logging Rates

You can enable logging of collected data into the resource usage tables. In this case, you also set a logging rate that specifies how often a data row is to be inserted.

Logging is not required to collect and retrieve current data, but it does preserve data that would otherwise be lost at the next collection interval.

If you decide to enable logging, set the logging rate. The resource logging rate, also referred to as the physical or virtual resource logging rate, sets the interval in seconds at which resource usage data is written to the resource usage tables (see [Data Collection](#)). The rate is saved on disk every time it is altered. Use the SET RESOURCE RATE request to set this rate (see [SET RESOURCE RATE](#)).

The resource logging rate:

- Can be set within the range of 0 and 3600 seconds.
- With a value of zero indicates that logging is not being performed.
- Returns the value ResLogging in the MONITOR PHYSICAL SUMMARY and MONITOR VIRTUAL SUMMARY requests.

---

#### Note:

A physical resource logging rate is the same as a virtual resource logging rate.

---

# Workload Management API Features and Examples

The following describes the functionality and features of the workload management API, which consists of interfaces to System PMPC, Teradata Dynamic Workload Management, Query Band, and embedded services system.

## System PMPC API Features

### Types of Tasks

The following types of tasks are tracked in Resource Usage tables:

- [Data Collection](#)
- [System-Level Monitoring](#)
- [Session-Level Monitoring](#)
- [Monitor Locks](#)

### Data Collection

System PMPC requests, except MONITOR VERSION and MONITOR SQL, are based on periodic data collection. The Resource Sampling Subsystem (RSS) rates at which the resource data is gathered (collection rate) and written to the resource usage tables (logging rate) are set separately.

You can control the session collection rate, the resource collection rate, and the resource logging rate. You can set the resource collection rate for any interval between 0 and 3600 seconds. You can set the session collection rate for any interval between 1 and 3600 seconds. For other rules governing logging rates, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

A single master resource collection system within the database collects all performance monitoring data. It can be accessed in a number of ways, such as using PM/API requests or SQL interfaces. Collection rates that are set this way can be reset by using the Database Window utility or Teradata Viewpoint.

---

#### Note:

You cannot set the session rate to zero in SET SESSION RATE PM/API or SetSessionRate Open API; however, it can be set to zero in the Supervisor Window.

---

Care should be taken to integrate the various performance monitoring tasks on your system to avoid potential conflicts.

---

#### Note:

Because resource usage data is collected in different memory repositories than session-level data, changes in the resource collection rate have no impact on session-level usage data, and vice versa.

---



The following table describes the various types of monitoring rates that are set using the following APIs:

- The SET RESOURCE RATE and SET SESSION RATE requests.
- The SetResourceRate and SetSessionRate functions.

Rate	Description
Global session (SesMonitorSys) monitoring	<p>Sets the maximum acceptable age of collected session-level data in memory to the PM/API application or end user.</p> <p>This rate is returned as SesMonitorSys value in a MONITOR VIRTUAL SUMMARY request.</p> <p>The global session rate impacts all MONITOR SESSION requests unless local session rate is set.</p>
Local session (SesMonitorLoc) monitoring	<p>Sets the maximum acceptable age of collected session-level data in memory for an individual Monitor partition session that submits a MONITOR SESSION request.</p> <p>This rate is returned as SesMonitorLoc value in a MONITOR VIRTUAL SUMMARY request.</p> <p>By default the local session rate is the same as the global session rate.</p> <p><b>Note:</b></p> <p>A change to the local collection rate could affect the cumulative data that other users see because all session usage data is stored in the same memory repository. Because changes to either the global or local rate can reset the starting point at which data is collected and may alter cumulative session usage data, it is important to restrict the granting of session monitoring privilege to users trained in the use of system monitoring tools, for example, the system or database administrator or certain application programmers.</p> <p>This rate is not saved on disk and is lost during a system outage.</p>
Resource monitoring (ResMonitor)	<p>Sets the interval in seconds at which all resource usage data is collected within memory for reporting via the PM/API.</p> <p>The resource monitoring rate is returned as a ResMonitor data value in a MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY request.</p> <p>You can use the SampleSec field of MONITOR PHYSICAL RESOURCE to view the current rate. This field is equivalent to the ResMonitor field.</p>
Resource logging (ResLogging)	<p>Sets the interval in seconds at which resource usage data is written to the resource usage tables.</p> <p>The resource logging rate is returned as a ResLogging data value in a MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY request.</p>

- Data collection rates must be set to a nonzero value for all data fields called by a PM/API request or SQL interface or the fields will not contain any data.
- Because all rates, except for the local session monitoring rate, are saved on disk every time they are altered, they are “remembered” during restarts.

## Related Information

For ...	See ...
information on global and local rates	<ul style="list-style-type: none"> <li>• <a href="#">SET SESSION RATE</a>.</li> <li>• <a href="#">MONITOR VIRTUAL SUMMARY</a>.</li> </ul>
information on resource monitoring and logging rates	<ul style="list-style-type: none"> <li>• <a href="#">SET RESOURCE RATE</a>.</li> <li>• <a href="#">MONITOR PHYSICAL SUMMARY</a>.</li> <li>• <a href="#">MONITOR VIRTUAL SUMMARY</a>.</li> </ul>
comparative information on setting logging rates	<ul style="list-style-type: none"> <li>• Control GDO Editor (ctl) information in <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</li> <li>• <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</li> <li>• <i>Teradata® Viewpoint User Guide</i>, B035-2206</li> </ul>
comparative information on setting collection rates	<ul style="list-style-type: none"> <li>• Database Window (xdbw) information in <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</li> <li>• <i>Teradata® Viewpoint User Guide</i>, B035-2206.</li> </ul>

## System-Level Monitoring

Use System PMPC to perform two types of system monitoring:

- Physical resources
  - Nodes availability
  - BYNET availability
- Virtual resources (vprocs)
  - Access Module Processor (AMP) status, performance and utilization
  - Parsing Engine (PE) status, performance and utilization

Resource utilization data is collected and reported differently from session utilization data. Whereas some of the session usage data is collected cumulatively, resource data is collected for a particular collection period. The resource data reported is based on the activity that occurred during that collection period and does not include any cumulative data over collection periods. For example, if you set the resource usage collection interval to 60 seconds and issue a MONITOR VIRTUAL RESOURCE request (or a MonitorVirtualResource function) or MONITOR PHYSICAL RESOURCE request (or MonitorPhysicalResource function), a report is issued for that specific 60-second interval.

Any data you do not examine within the 60 seconds is lost when it is overwritten by data collected during the next 60-second collection interval.

Resource usage data and session-level usage data are deposited in separate global data collection areas. The data in the repository is updated once each collection period. All users share the data, which is used to generate responses.

## Session-Level Monitoring

Session-level monitoring tasks return the following information:

- Identification of blocking users, sessions and locked databases or tables
- Session-level usage data on:
  - AMPs
  - CPUs
- Identification of problem SQL requests, including:
  - Current session
  - Current step
  - SQL text EXPLAIN data

Some of the session-level utilization data is collected cumulatively. The session rate is used to limit the frequency at which cumulative data is updated. For example, if you set the session rate to 60 seconds and issue a MONITOR SESSION request every 60 seconds, session-level usage data and request-level usage data is cumulatively totaled and updated every 60 seconds. Cumulative type session-level or request-level data reported includes data from the beginning of the session or request.

## Monitor Locks

Locks may occur when sessions, utilities, and applications being run by specific users block access to databases or tables normally available from the database. Interfaces to System PMPC can help you monitor locks.

To help determine the user causing a block and the locked database or table, you can use the MONITOR SESSION request or the MonitorSession function. Then, to get more specific information about the blocking session and the object being blocked, you can use the IDENTIFY request or IdentifySession, IdentifyUser or IdentifyTable functions.

To learn more about the interfaces used to perform these functions, see [System PMPC APIs](#).

## Examples of Job Control Support Applications Using PM/APIs

This section explains two advanced examples of potential job control support applications that use PM/API requests:

- Resource Supervisor
- Idle Session Logoff

They are explained at a high level so that, by understanding the concepts, you can develop similar applications at a customer site for monitoring and controlling the use of database resources.

### Resource Supervisor

A Resource Supervisor prevents runaway queries. Runaway queries are sometimes a problem at a site where end users can access the database to make ad hoc Teradata SQL requests. A badly formulated query (for example, one missing constraint on a WHERE clause) could inadvertently cause a product

join, which consumes more resources than the user intended. Further, a user making an ill-formed SQL statement might request a join on two big tables, which unintentionally results in a Cartesian product join. The Resource Supervisor aborts transactions that exceed a certain resource usage threshold.

You can write a Resource Supervisor to use features available in the request as shown in the example below.

1. Program the SET SESSION RATE request to set a reasonable session-level collection rate, for example, 10 minutes.
2. Based on the session-level rate, program the client application to issue a MONITOR SESSION request for all sessions or for a subset of sessions (for example, if users from a specific client are the only ones to be governed).
3. For each session returned to the client, program the client application to check some site-specific criteria to see if the session is a candidate for the Resource Supervisor.

For example, interactive users are required to have INTERACTIVE as the first word of their account string. If only interactive users are to be monitored by the Resource Supervisor, all sessions that do not include INTERACTIVE as the first word of the UserAccount value returned by a MONITOR SESSION request are ignored.

4. For those sessions that are candidates for the Resource Supervisor, program the client application to look at the AMPCPUSec, PECPUSec, and AMPIO values to determine if some site-specific maximum acceptable value has been exceeded.

These session values are cumulative and may not be appropriate for use as a Governor limit because you are limiting the total resource usage of a session and not of a request.

5. Program the client application to keep a history of all previous cumulative values of AMPCPUSec, PECPUSec, and AMPIO, plus current XactCount (transaction count) and ReqCount (request count) values for each session.

The difference between the historical value and the current value tells you the resources used. The request count and transaction count values tell you if they are consumed as part of the current transaction or as part of the new request.

6. If the Resource Supervisor determines that a particular session has exceeded site-specific limits, program the client application to issue an ABORT SESSION request for those session that have exceeded the limits.

The client application can specify the *logoff* option for the ABORT SESSION request, depending on how severely the offending session is controlled.

### Idle Session Logoff

An Idle Session Logoff application automatically logs off users whose sessions have been idle for a certain length of time. This job control support feature prevents users from walking away from a terminal and allowing unauthorized users access to sensitive information.

You can write an Idle Session Logoff application program using the requests described below.

1. Program the SET SESSION RATE request to set a reasonable session-level collection rate, for example, 10 minutes.

- Based on the session-level rate, program the client application to issue a MONITOR SESSION request for all sessions or for a subset of sessions (for example, if users from a specific client are the only ones to be monitored for idle sessions).

- For each session returned to the client, check some site-specific criteria to see if the session is a candidate for Idle Session Logoff.

For example, interactive users are required to have INTERACTIVE as the first word of their account string. If only interactive users are to be monitored by the Resource Supervisor, all sessions that did not have INTERACTIVE as the first word of the UserAccount value returned by a MONITOR SESSION request are ignored. Those sessions with the INTERACTIVE label proceed through the next step.

- For sessions that are candidates for Idle Session Logoff, program the client application to verify the following conditions to determine whether the session has been inactive for the duration of the collection period:

- AMPState and PESTate are idle.
- The session was idle during the last MONITOR SESSION request.
- XactCount and ReqCount values did not change during the last MONITOR SESSION request.

If all these conditions are met, the session has been inactive for the duration of the collection interval and is a candidate for automatic logoff.

- Program the client application to issue an ABORT SESSION request with the *logoff* option for the sessions that are candidates for automatic logoff.

## Examples Using Open APIs

The following table describes the different uses of the System PMPC open APIs.

You can ...	To ...
execute the MonitorSession and AbortSessions functions	create a query that aborts queries submitted by a set of users that have been running longer than 10 minutes and have been skewed by more than 30% for 20 minutes.
execute the MonitorSession and SetSessionAccount functions	change the account string.
execute the MonitorSession or MonitorSQLText functions	display the SQL of all active sessions that have run over 20 minutes.
select the blocked fields of the MonitorSession function	display the block information of all blocked sessions.

## Functionality

The following table describes the System PMPC interfaces that are used to show how efficiently the database is using its resources, to identify problem sessions and users, and to abort sessions and users having a negative impact on system performance.

If you want to ...	Use the following SQL interface. ..	OR use the following CLIV2 or Teradata JDBC Driver request ...
abort any outstanding requests or transactions of one or more sessions	<a href="#">AbortSessions</a> or <a href="#">AbortListSessions</a>	<a href="#">ABORT SESSION</a>
return the name of a user, by session, who is causing a block	<a href="#">IdentifySession</a>	<a href="#">IDENTIFY</a>
return the name of the specified table ID	<a href="#">IdentifyTable</a>	<a href="#">IDENTIFY</a>
return the name of the specified user ID who is causing a block	<a href="#">IdentifyUser</a>	<a href="#">IDENTIFY</a>
collect statistics on AMPs based on the in-use AMP Worker Tasks (AWTs)	<a href="#">MonitorAMPLoad</a> or <a href="#">MonitorAWTResource</a>	<a href="#">MONITOR AWT RESOURCE</a>
collect session information for the current user on the current host	<a href="#">MonitorMySessions</a>	—
collect overall information on node availability	<a href="#">MonitorPhysicalConfig</a>	<a href="#">MONITOR PHYSICAL CONFIG</a>
collect RSS data and returns node-specific data	<a href="#">MonitorPhysicalResource</a>	<a href="#">MONITOR PHYSICAL RESOURCE</a>
collect global summary information	<a href="#">MonitorPhysicalSummary</a>	<a href="#">MONITOR PHYSICAL SUMMARY</a>
return session or request resource usage statistics	<a href="#">MonitorSession</a>	<a href="#">MONITOR SESSION</a>
return session rate	<a href="#">MonitorSessionRate</a>	<a href="#">MONITOR SESSION</a>
return data about the step being executed of the currently running request	<a href="#">MonitorSQLCurrentStep</a>	<a href="#">MONITOR SQL</a>
return the step information of the current or running request	<a href="#">MonitorSQLSteps</a>	<a href="#">MONITOR SQL</a>
return the SQL text of the request currently being executed for the specified host, session, and vproc	<a href="#">MonitorSQLText</a>	<a href="#">MONITOR SQL</a>
return BYNET status and system type values that are generated once for the entire system or collect overall information on node availability	<a href="#">MonitorSystemPhysicalConfig</a>	<a href="#">MONITOR PHYSICAL CONFIG</a>
collect information on virtual processor (vproc) availability	<a href="#">MonitorVirtualConfig</a>	<a href="#">MONITOR VIRTUAL CONFIG</a>

If you want to ...	Use the following SQL interface. ..	OR use the following CLIV2 or Teradata JDBC Driver request ...
collect performance information for each AMP, PE, or TVS vproc	<a href="#">MonitorVirtualResource</a>	<a href="#">MONITOR VIRTUAL RESOURCE</a>
collect global summary information on system utilization	<a href="#">MonitorVirtualSummary</a>	<a href="#">MONITOR VIRTUAL SUMMARY</a>
return ResUsageSps data from the RSS SPS memory buffer	<a href="#">MonitorWD</a>	<a href="#">MONITOR WD</a>
return a subset of the RSS ResUsageSps data or return the collection rate, number of nodes with at least one online AMP, and number of nodes with at least one online PE	<a href="#">MonitorWDRate</a>	<a href="#">MONITOR WD</a>
set either the: <ul style="list-style-type: none"> <li>• ResMonitor rate</li> <li>• ResLogging rate</li> </ul>	<a href="#">SetResourceRate</a>	<a href="#">SET RESOURCE RATE</a>
change the account string for the session or for the request.	<a href="#">SetSessionAccount</a>	<a href="#">SET SESSION ACCOUNT</a>
set the global and local rates for updating session-level statistics in memory	<a href="#">SetSessionRate</a>	<a href="#">SET SESSION RATE</a>

## Teradata Dynamic Workload Management API Features

Teradata dynamic workload management software is a rule-oriented management system capable of detecting and acting on events. It is a key component of Teradata Active System Management (TASM).

### Note:

Teradata ASM is not fully supported on Teradata appliances. For more information, see appropriate appliance documents for further details.

Teradata Dynamic Workload Management APIs allow database administrators to:

- Apply updates to the TASM rule categories (For descriptions of these rules, see *Teradata® Viewpoint User Guide*, B035-2206.)
- Monitor delayed requests
- Release or abort requests from the delay queue
- Display the workload definition of a query
- Determine the current status of TASM rules and rule sets
- Review statistics on how TASM rule categories are affecting request processing

- Retrieve information about the current status of all event-related constructs
- Enable or disable user-defined events for event management

For information on TASM rules and how to enable the rule categories in the Teradata Viewpoint Workload Designer portlet, see *Teradata® Viewpoint User Guide*, B035-2206.

For examples on performing these Teradata Dynamic Workload Management functions, see the following examples.

## Examples Using Open APIs

The following table describes the different uses of the Teradata Dynamic Workload Management open APIs.

You can execute the ...	To ...
MonitorSession and TDWMAssignWD functions	change the workload to WD-Report-High of all active requests with a workload of WD-Report-Low.
TDWMEventControl function	change the active health condition or planned environment on the system to adjust the throttles.
TDWMEventStatus function	display all active events.
TDWMGetDelayedQueries and TDWMReleaseDelayedRequest functions	release all delayed queries for a specified workload.
TDWMRuleControl function	temporarily enable a rule to block an application for accessing a database while it is synchronized between two active systems.
TDWMSummary function	display the current workload summary statistics.
TDWMThrottleStatistics function	display the current delay queue statistics.

## Functionality

The following table describes the interfaces of the Teradata Dynamic Workload Management that are used to return WDs, delayed query lists, summary data, and statistics; and update the components stored in the Teradata Dynamic Workload Management database.

### Note:

The updates in the TDWMRuleControl and TDWMSetLimits procedures must be committed before the TDWMAApply procedure can be called. If the same transaction is used to call the external stored procedures that update the TDWM database and the TDWMAApply external stored procedure, a self-deadlock occurs as shown in the example below.

```
Bt;
Call TDWMRuleControl()
```



The updates are not committed because the TDWMRuleControl procedure is in a transaction (that is, the rows are locked for write).

```
Call TDWMApplly(200, 'Y','N','N','N');
```

The TDWM database waits on the locked tables.

If you want to ...	Use the following SQL interface ...	Or the following CLIV2 or Teradata JDBC Driver request ...
abort a request or utility session on the Teradata dynamic workload management software delay queue	<a href="#">TDWMAbortDelayedRequest</a>	<a href="#">TDWM DELAY REQUEST CHANGE</a>
apply changes to the rules in one or more the Teradata dynamic workload management software categories	<a href="#">TDWMApplly</a>	—
change the workload a session or request is assigned to	<a href="#">TDWMAssignWD</a>	<a href="#">TDWM WD ASSIGNMENT</a>
activate or deactivate a user-defined event	<a href="#">TDWMEventControl</a>	<a href="#">USER EVENT CONTROL</a>
list all objects that make up the system state	<a href="#">TDWMEventMapping</a>	<a href="#">EVENT STATUS</a>
return the currently defined events	<a href="#">TDWMEventStatus</a>	<a href="#">EVENT STATUS</a>
return the collection rate (that is, the first record of the Teradata dynamic workload management software Exception PM /API request) or return Teradata dynamic workload management software exception data from the database	<a href="#">TDWMExceptionRate</a>	<a href="#">TDWM EXCEPTIONS</a>
collect the Teradata dynamic workload management software exception data from the database	<a href="#">TDWMExceptions</a>	<a href="#">TDWM EXCEPTIONS</a>
report if each category is active or not	<a href="#">TDWMInquire</a>	—
return the delayed query data fields and delay information	<a href="#">TDWMGetDelayedQueries</a>	<a href="#">TDWM STATISTICS</a>
return the utility delay queue	<a href="#">TDWMGetDelayedUtilities</a>	<a href="#">TDWM STATISTICS</a>
return a list of the WDs	<a href="#">TDWMListWDs</a>	<a href="#">TDWM LIST WD</a>
return the statistics on the load utilities that are available in the system	<a href="#">TDWMLoadUtilStatistics</a>	<a href="#">TDWM STATISTICS</a>

If you want to ...	Use the following SQL interface ...	Or the following CLIv2 or Teradata JDBC Driver request ...
release a request or utility session on the Teradata dynamic workload management software delay queue	<a href="#">TDWMReleaseDelayedRequest</a>	<a href="#">TDWM DELAY REQUEST CHANGE</a>
return the Teradata dynamic workload management software WD summary data fields	<a href="#">TDWMSummary</a>	<a href="#">TDWM SUMMARY</a>
return the collection rate	<a href="#">TDWMSummaryRate</a>	<a href="#">TDWM SUMMARY</a>
return statistics for throttled database objects or throttled workloads	<a href="#">TDWMThrottleStatistics</a>	<a href="#">TDWM STATISTICS</a>

## Query Band API Features

Query banding is a method for tracking system usage and managing task priorities. A query band is a list of “name=value” pairs in a string contained within apostrophes that is defined by the user or middle-tier application as shown below.

```
'org=Finance;report=EndOfYear;universe=west;'
```

### Note:

The name-value pairs are separated by a semicolon.

There are three types of query bands:

- A session query band, which is stored in the session table and recovered after a system reset.
- A transaction query band, which is discarded when the transaction ends (for example, a commit, rollback, or abort).
- A profile query band, which is set for the session at logon. The profile query band is not saved in the session table, so after a restart the session is initialized with the profile query band based on the current profile setting.

You can set a query band for the transaction and session using the SQL statement, SET QUERY\_BAND. For information on SET QUERY\_BAND, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

You can set a default query band in a profile with the CREATE PROFILE statement and assign it to a user with CREATE USER or MODIFY USER. For information on CREATE PROFILE, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

By setting a query band, you can:

- Identify the user, application, or report that originated the request from a middle-tiered application.

- Identify what user, application, report, and even what part of an application issued a request (for example, a query band can be used for accounting, troubleshooting, and in other types of system management operations).
- Give requests a higher priority. For example, a query band can make a request issued for an interactive report a higher priority than one issued for a report that generates output files.
- Increase the priority of an urgent job. For example, if the CEO needs a report for a board review that starts in 20 minutes, a query band can be used to expedite the job.
- Create requests that make up a “job” to be grouped for accounting and control purposes.

There are several uses for query bands. A query band can be:

- Logged by Database Query Log (DBQL). DBQL reports are created using the query band name-values pairs to provide additional refinement for accounting and resource allocation purposes and to assist in troubleshooting performance problems.
- Used for rule checking and Workload Classification. Query band name-value pairs can be associated with TASM Filter rules and defined as workload attributes (see *Teradata® Viewpoint User Guide*, B035-2206 for details on these rules).
- Used to determine the origin of a request that may be consuming system resources or blocking other requests.
- Used as a system variable. A query band can be set for a session and retrieved using APIs.

Through these interfaces, the following information can be retrieved:

- The concatenated transaction and session query band for the specified session.
- The concatenated query band for the current transaction and session.
- The name and value pairs in the query band.
- The value of the specified name in the current query band.

For examples on performing Query band requests and functions, see [Examples Using PM/API and Open APIs](#).

To learn more about these interfaces and how to retrieve query bands, see [Workload Management: Query Band APIs](#).

## Examples Using PM/API and Open APIs

The following table describes the different uses of the query band APIs.

You can use ...	To ...
MONITOR QUERYBAND or MonitorQueryBand	return the concatenated query band for session number 1102 on host ID 20 running on vproc 16383.
GetQueryBand or GetQueryBandSP	return the concatenated query band string for the current transaction, session, and profile.
GetQueryBandValue	query the DBQLLogTbl based on names and values specified in the query band name input argument.

You can use ...	To ...
<code>GetQueryBandValueSP</code>	search the session name-value pairs in the query band and return the value that corresponds to the query band name "aa."
<code>GetQueryBandPairs</code>	return the query band in name and value columns.

## Functionality

The following table describes the query band interfaces that are used to track system usage and manage task priorities.

If you want to ...	Use the following SQL interface ...	Or, the following CLIV2 or Teradata JDBC Driver request ...
return the name and value pairs in the query band	<a href="#">GetQueryBandPairs</a>	—
return the concatenated query band for the current transaction and session	<a href="#">GetQueryBand</a> or <a href="#">GetQueryBandSP</a>	—
return the value of the specified name in the current query band or NULL	<a href="#">GetQueryBandValue</a> or <a href="#">GetQueryBandValueSP</a>	—
return the concatenated query band for the specified session	<a href="#">MonitorQueryBand</a>	<a href="#">MONITOR QUERYBAND</a>
return all query band names and descriptions, including those dropped from a release	<a href="#">QueryBandReservedNames_TBF</a>	—

## Embedded Services System API Features

### Functionality

Embedded services system APIs follow the implicit data type conversion rules that apply to UDFs. For more information about the implicit data type conversion rules, see *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

The following table describes the different embedded services system interfaces.

If you want to ...	Use the following SQL interface ...
return the current type of Priority Scheduler being used (for example, workload definitions)	<a href="#">GetPSFVersion</a> .

If you want to ...	Use the following SQL interface ...
retrieve the maximum CPU and I/O capacity on demand (COD) values from the Priority Scheduler	<a href="#">TD_get_COD_limits.</a>
return the value of the specified name in the current query band or NULL	<a href="#">GetQueryBandValueSF.</a>

# System PMPC APIs

The following discusses how to use the System Performance Monitor and Production Control (PMPC) APIs to monitor system and session-level performance and to identify and abort problem sessions and users having a negative impact on system performance.

PMPC data is available through two types of interfaces:

- [PM/APIs](#)
- [Open APIs \(SQL Interfaces\)](#)

Before using the System PMPC APIs, you may want to familiarize yourself with [Examples of Job Control Support Applications Using PM/APIs](#).

## PM/APIs

This section describes the Teradata Dynamic Workload Management interfaces that use CLIV2 or the Teradata JDBC Driver. These interfaces are referred to as PM/API requests.

For details on using the Teradata JDBC Driver requests, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

Before using the System PMPC PM/APIs, read the following topic on the impact of object name length on PM/API requests.

### Impact of Object Name Length on PM/API Requests

Object names contain 128 Unicode characters; however, to use object names of 128 characters, applications must use monitor software version 10 or later.

If your custom application uses monitor software version ...	The object field in the input area can be ...
8 or earlier	CHAR(30). A fixed parcel field size of 30 bytes. The names must be padded with spaces. For more information on monitor software version 8 or earlier, see previous releases of <i>Workload Management API: PM/API and Open API</i> .
9	VARCHAR(120). A parcel field size of up to 120 bytes in variable length. For more information about monitor software version 9, see <i>Application Programming Reference: Workload Management</i> .
10 or later	VARCHAR(512). A parcel field size of up to 512 bytes in variable length in host character set format.

Depending on the version of your custom application, some of the PM/API request output fields for object names may be truncated. For example, if the output field is an object name of 100 bytes in length when converted to host character set format, and you are using a PM/API request with monitor software version 8 or earlier, the database name will be truncated to fit the fixed parcel field size of 30 bytes.

## Common Warning and Error Messages

Most of the monitor requests covered in this section depend on periodic collection of data. When this collection process is disturbed, the returned data can temporarily become confusing or inaccurate.

If this happens, the user receives a warning message indicating that the data returned may not give a true picture of system or user activity.

The following are some common situations in which a warning message is provided:

- The data returned may have been affected by a system restart or a single processor recovery. Resource/session usage values may not be realistic until requests on the database re-attain their normal level of activity.
- Data may also be incomplete in cases where:
  - A full data collection period has not transpired since the database has gone through a recovery.
  - A full data collection period has not transpired since the related monitoring rate was changed.

For more detailed information on warning and error messages, see *Teradata Vantage™ - Database Messages*, B035-1096.

## Errors Resulting from PMPC Subsystem Failures

PMPC fault isolation provides automatic recovery from internal software errors or faults in System PMPC tasks. This significantly reduces the cases in which the PMPC subsystem must be shut down because of internal software errors or faults in PMPC modules. The PMPC subsystem continues to be available after clean up of local resources, in most cases.

When an internal software error or a fault occurs in PMPC tasks, all current MONITOR partition sessions are forced off with an error and logons to the MONITOR partition are temporarily disabled while the PMPC subsystem recovery is in progress. After the recovery is complete the logons to the MONITOR partition are re-enabled. The PMPC subsystem recovery is expected to take up to 3 minutes on a system with a normal workload.

---

### Note:

Only the PMPC subsystem restarts on an internal software error or a fault in PMPC tasks. The requests in DBC/SQL sessions and all other type of non-PMPC requests are not impacted or interrupted while the PMPC subsystem recovery is in progress. Also a snapshot dump may have been taken during the recovery. The administrator should report the snapshot dump and any related errors in the event log to the Teradata Support Center.

---

## ABORT SESSION

Aborts any outstanding request or transaction of one or more sessions, and optionally logs those sessions off the database system.

### Input Data

If an ABORT SESSION request is submitted when *host\_id*, *user\_name*, or *session\_no* is either left blank or set to NULL, an attempt is made to abort all hosts, all users, or all sessions. For example, if you specify 127 for *host\_id*, FRR for *user\_name*, and NULL for *session\_no*, ABORT SESSION attempts to abort every and all sessions currently logged on from host 127 as user FRR.

Element	Data Type/ Range	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .
<i>host_id</i>	SMALLINT	Logical ID of a host (or client) with sessions logged on. For example, a <i>hostid</i> of zero identifies internal sessions or system console sessions. <i>host_id</i> cannot exceed 1023.
<i>session_no</i>	INTEGER	Session number. <i>session_no</i> combined with the <i>host_id</i> represents a unique session ID.
<i>user_name</i>	VARCHAR(128)	Name of user who is running the sessions.
<i>list</i>	VARCHAR(1)	Indicator of whether to display a list of sessions. To display a list of all <i>session_no</i> sessions, specify y or Y. If you do not want to display a list of sessions, specify n, N, NULL, or blank. Do not use <i>list</i> when large numbers of sessions are being aborted. Otherwise, system degradation can result, especially when the abort list contains more than 1500 sessions. The slowdown occurs because all of the impacted sessions must be buffered and then sorted on a single processor. During the sort, the processor is unavailable for requests from any other MONITOR session logged on to that PE. Furthermore, in extremely rare cases, perhaps involving an abort of 10,000 sessions, the number of sessions needed to be sorted can exhaust scratch space on the processor and cause a system restart.



Element	Data Type/ Range	Description
		<p><b>Note:</b></p> <p>If your site is running Two-Phase Commit (2PC), the following information applies. For more information on 2PC, see <i>Teradata Vantage™ - Database Design</i>, B035-1094.</p> <ul style="list-style-type: none"> <li>Do not use ABORT SESSION to abort or log off Teradata Director Program internal sessions used for 2PC processing. To log off the sessions, use the Teradata Director Program command DISABLE IRF instead.</li> <li>Be sure to issue DISABLE IRF before executing ABORT SESSION using the <i>host_id.*</i> or <i>.*</i> parameters to avoid any possible problem with the Teradata Director Program.</li> <li>To identify Teradata Director Program internal sessions, run the Teradata Director Program command DISPLAY SESSIONS. Teradata Director Program internal sessions have the job name *TDPINT*. (The MONITOR SESSION request cannot identify Teradata Director Program internal sessions.)</li> </ul>
<i>logoff</i>	VARCHAR(1)	<p>Indicator of whether to log <i>session_no</i> sessions off the database in addition to aborting them.</p> <p>To log <i>session_no</i> sessions off the database, specify y or Y. To abort <i>session_no</i> sessions, specify n, N, NULL, or blank.</p>
<i>override</i>	VARCHAR(1)	<p>Possible values:</p> <ul style="list-style-type: none"> <li>y or Y. If you specify y or Y, you do not want the ABORT SESSION request to fail in any of the following cases: <ul style="list-style-type: none"> <li>An identified session is being session-switched.</li> <li>An identified session is executing its own ABORT SESSION request.</li> <li>An identified session has a PESTate of IDLE: IN-DOUBT as a result of a 2PC.</li> </ul> </li> </ul> <p><b>Note:</b></p> <p>Sessions are marked IN-DOUBT by the 2PC protocol, which governs how transactions are committed by multiple systems that do not share memory. The protocol guarantees that either all systems commit or all roll back.</p> <p>Therefore, when you specify y or Y, <i>session_no</i> sessions that fit one of the above criteria are ignored, and all sessions that do not fit any of the above criteria are aborted.</p> <ul style="list-style-type: none"> <li>n, N, NULL, or blank. If you specify n, N, NULL, or blank, and at least one identified session fits one of the above criteria, the ABORT SESSION request will fail.</li> </ul> <p>If you specify n or N, or do not specify this option, the entire ABORT SESSION operation fails if any non-abortable session is encountered. If sessions are not aborted (including those in abortable states), this field returns an error message.</p>

## Performance and the *list* Option

The ABORT SESSION request is different from other PM/API requests, because as soon as processing begins, the ABORT SESSION operation **is not abortable**.

The ABORT SESSION cannot be stopped even if *list* displays some sessions that were included by mistake.

## Monitor Privileges

To use this request, you must have the ABORTSESSION privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - ABORT SESSION

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

Aborting a session is useful when a session causes a production job to block other sessions waiting for locks, or when a session takes up many resources that a critical application is running too slowly.

By default, the ABORT SESSION request aborts the current transaction for the specified sessions and logs those sessions off the database.

The ABORT SESSION request is logged to the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view). If you specify y or Y for *logoff*, records are added to DBC.SW\_Event\_Log table.

ABORT SESSION will not abort nor log off a session under any of the following conditions:

- The session status is IN-DOUBT.
- The session is currently being session-switched.
- The session is a Database Query Log (DBQL) artificial internal session
- The actual session is currently executing an ABORT SESSION request.

If you attempt to abort a session that is currently executing an ABORT SESSION request, either a diagnostic message displays indicating that the ABORT SESSION request was not accepted, or, the ABORT SESSION request aborts any identified sessions that do not meet the above conditions.

Although this request may abort and log off other PM/API sessions, it ignores the session from which it was submitted.

At least one of the transactions you want to abort either cannot be aborted or already is being aborted if:

- A session is in the final stage of an ALTER TABLE operation and cannot be aborted.

- A user-initiated abort must complete before whatever caused the ABORT SESSION request to be executed can be done. Therefore, if you specify *list*, that list will indicate sessions that are:
  - Currently IN-DOUBT
  - Being session-switched
  - Already aborting or committing their own transactions
  - Executing an ABORT SESSION request

The ABORT SESSION request has the following impact as described in the table below.

Type of Session	Impact of ABORT SESSION	Option	Comments
A session with a transaction that is currently being committed or rolled back	None	with or without <i>logoff</i>	The ABORT SESSION request does not fail. Instead, the stage 1 response from the database includes a warning that identified sessions are in the process of committing or rolling back their transactions.
An internal session	None		<p>The activity of such sessions is vital to the continued database execution and is always considered to be more important than any user-initiated work that may be blocked. The database system acts as if the internal sessions do not appear in the optional list of sessions identified by this request.</p> <p><b>Note:</b> Specify a <i>host_id</i> of zero to abort console Basic Teradata Query (BTEQ) sessions.</p>
A DBQL/Teradata dynamic workload management software artificial internal session	Does not work		This is not a real session and cannot be aborted. If you attempt to abort this session, an error message is returned.
A client utility user	Little		Client utility locks are designed to survive system outages or interruptions in the archive process, and are not necessarily associated with an active session. Instead, they are associated with the user who originally submitted the archive or recovery operation. Therefore, an ABORT SESSION request does not necessarily remove locks placed by the client utility and does not necessarily cause whatever activity any such locks were blocking to become unblocked. However, such sessions still appear in the optional list of sessions identified by this request.
MLOAD, FASTLOAD, and /or DBCUTIL partition sessions	Does not work	without <i>logoff</i>	Sessions within these partitions do not have transactions or locks associated with them. Also, many of these partitions do not include the Teradata SQL or PM/API ability to recover from an interrupted request without terminating the application. That is, these partitions are designed to be restarted, but not rolled back. As a result, when a FastLoad or MultiLoad operation needs

Type of Session	Impact of ABORT SESSION	Option	Comments
			<p>to be terminated, the ABORT SESSION request typically specifies that all sessions associated with the utility job be aborted and logged off. These utility-related sessions appear in the optional list of sessions identified by this request.</p> <p>It is easier to kill a FastLoad, MultiLoad, or FastExport job by <i>user_name</i> instead of by <i>session_no</i>. These utilities have multiple sessions under one <i>user_name</i> and it makes little sense to abort one utility session running under that <i>user_name</i> and not another. However, be aware that the same <i>user_name</i> may be running other sessions at the same time and these sessions would also be aborted and logged off.</p>

## Processing Messages

Unlike other PM/API requests, ABORT SESSION has an unpredictable execution time and could take hours to roll back a transaction.

Therefore, upon receipt of an ABORT SESSION request, the database sends one or more of the following processing messages:

- Indicates that the ABORT SESSION request is received and is in one of the following states:
  - Is accepted and is in execution.
  - Cannot be accepted because resources are exhausted, because of either a heavy workload or several ABORT SESSION requests are queued and waiting to finish processing. Information is recorded in the error log.

The database should not restart as a result of this error. Depending on the available resources, this request will be processed to completion after all the queued abort requests have completed processing.

- Lists how many sessions have been affected by the executing request (if Y was entered into the *list* field).
- Indicates the request is complete when all identified sessions have been aborted.
- Indicates that all session have been logged off (if Y was entered into the logoff field).

## CLv2 Response Parcels

Because of the unpredictable execution time of ABORT SESSION, the database system handles ABORT SESSION as if it were a two-statement request, with each statement generating a response.

The two-statement response contains the following sequence of parcel types.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = Number of sessions identified by the ABORT SESSION request ActivityType = 86 (PCLABTSESS)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Data or IndicData list of sessions. This parcel is present only if you specified <i>list</i> .
EndStatement	11	6	StatementNo = 2-byte integer.
Success	8	18 to 273	StatementNo = 2 ActivityCount = 0 (Number of sessions identified by the ABORT SESSION request) ActivityType = 86 (PCLABTSESS)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
EndStatement	11	6	StatementNo = 2-byte integer.
EndRequest	12	4	None.

The only difference between the parcels returned with the *list* option set to y or Y, and those returned with the *list* option set to n, N, blank, or NULL, is that one or more Record parcels are added when you specify the former.

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

In the first response, the database indicates that the request was accepted and is being executed. It also lists the affected sessions and their status if you specify y or Y for *list*.

### Statement 1

The Record parcel in the first statement of the response returns the list of sessions in increasing order of HostId. The following table shows the values returned from the first statement.

Field/ Column Name	Data Type	Description
HostId	SMALLINT	Logical host ID associated with a PE or session. For a PE, the Host ID identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.  <b>Note:</b> This value is NULL for AMPs. A value of zero represents the Supervisor window.
UserName	VARCHAR(128) CHARACTER SET UNICODE NOT NULL	User name of the session.
SessionNo	INTEGER NOT NULL	Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.
AbortStatus	VARCHAR(1)	Information on the status of the associated session: <ul style="list-style-type: none"> <li>• I = In-Doubt</li> <li>• A = Aborting a transaction</li> <li>• C = Committing a transaction</li> <li>• E = Executing an ABORT SESSION request</li> <li>• S = Switching</li> <li>• NULL = In some state other than the above</li> </ul> For an ABORT without LOGOFF, any status except NULL indicates the reason the request could not impact the associated session. For an ABORT with LOGOFF, an I, E, or S status value indicates that the associated session cannot be aborted or logged off.

If you do not specify logging off sessions, the Success parcel produced for the first statement, when applicable, includes a warning that identified sessions are in the process of committing or rolling back their transactions. An ABORT with *logoff* completes normally and does not produce any warnings.

## Statement 2

In the second response, the database indicates that the request is complete. The completion response is sent only after all impacted sessions have been aborted and, if requested, logged off.

## Sample Input - CLlv2 Request

The following example illustrates how the parcels for an ABORT SESSION request, built by CLlv2, look when sent to the database server using a *host\_id* of 52, *session\_no* of 2521, *user\_name* of user\_01, *list* of Y, *logoff* of Y, and *override* of N. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	17	Request	ABORT SESSION
0003	Data	47	MonVerID	2
			HostId	348
			SessionNo	1000
			UserName	user_01
			ListOption	Y
			Logoff	Y
			Override	N
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

With a *host\_id* of 52, *session\_no* of 2521, *user\_name* of DBC, *list* of Y, and *logoff* of Y, this request might return the following values in character text format. Your application program may return the values in a different format or display.

```

Success parcel:
  StatementNo: 1    ActivityCount: 1
  ActivityType: 86   FieldCount: 4
DataInfo iparcel:
  FieldCount: 4
Record parcel.
  Parcel flavor:      10    Parcel body length:  38
  HostId = 52,  UserName = "DBC",  SessionNo = 2521,
  AbortStatus = ' '.
EndStatement.
Success parcel:
  StatementNo: 2    ActivityCount: 0
  ActivityType: 86   FieldCount: 0
DataInfo parcel:
  FieldCount: 0

```

```
EndStatement.
EndRequest.
```

## Relationship Between ABORT SESSION and MONITOR SESSION

When sessions are being aborted or sessions are being blocked by the sessions being aborted, data returned from subsequent MONITOR SESSION queries may be affected. After the abort operation starts, you can immediately notice the changes from aborting sessions. However, you will not notice the changes resulting from sessions that were blocked by aborting sessions in MONITOR SESSION responses until the abort operation is complete.

### Aborted Sessions

For aborted sessions, expect the following types of changes in the response returned from a subsequent MONITOR SESSION query:

- PEState changes to PARSING. AMPState changes to ABORTING.
- AMPCPUsec is updated due to its continued tracking of the CPU time used during the transaction rollback process. Resources consumed during a transaction rollback are charged to the user the same way as any other form of resource usage.
- AMPIO is updated due to its tracking of logical I/O necessary during the transaction rollback process.
- Request\_AMPSPool decreases if there are pool files in use by the aborted request, because those pool files are discarded.
- After the abort operation has completed and if the aborted sessions are not logged off, the sessions become IDLE (PEState) while they wait for subsequent requests.

If the aborted sessions are logged off, they are no longer tracked by subsequent MONITOR SESSION requests.

### Blocked Sessions

For sessions blocked or slowed down by aborted sessions, expect the following types of changes in the response returned from a subsequent MONITOR SESSION query:

- The following response fields related to the aborted sessions are removed (that is, they are reported as NULLS because the blocking session is gone):
  - Blk\_x\_HostId
  - Blk\_x\_SessNo
  - Blk\_x\_UserID
  - Blk\_x\_LMode
  - Blk\_x\_OLType
  - Blk\_x\_ObjDBId
  - Blk\_x\_ObjTId
  - Blk\_x\_Status

For example, the following describes the information that can be inferred from the returned data:



- Within a MONITOR SESSION, response fields show Session 1 blocked by Sessions 2 and 3.
- After an ABORT SESSION, Session 2 is removed.
- Within a subsequent MONITOR SESSION, response fields show Session 1 blocked by Session 3.

- The MoreBlockers field may return data indicating there are no additional lock conflicts.

Because removing an aborted session allows another blocking session to be reported, there may be no remaining locks to report.

- If the aborted sessions are no longer blocking other sessions, the AMPState of those other sessions changes from BLOCKED to ACTIVE.
- For sessions that have changed to ACTIVE, or that are only slowed down by the aborted sessions, all resource usage fields may show a more rapid increase in resource usage. For example, the AMPCPUSec, AMPPIO, and Request\_AMPSPool fields may change more rapidly due to reduction in competition for those resources.

### **Before You Execute the ABORT SESSION Request**

Execute the MONITOR SESSION before you execute the ABORT SESSION request to get a list of current sessions. Therefore, you can evaluate which sessions to abort. You can use the host ID, session number, and user name returned by the MONITOR SESSION request as input data to an ABORT SESSION request.

Some of the values returned by MONITOR SESSION can help you determine which session is not actively processing or has a long-running transaction. For example, look at PESTate for the session in question. If PESTate is not PARSING-WAIT, PARSING, DISPATCHING, BLOCKED, or ACTIVE, that session may be a good candidate for an abort. As another example, look at the ratio of AMPCPUSec to XactCount to determine which transaction is running for a long time. If this ratio is high, this session has a long-running transaction.

It is also useful to run the QUERY SESSION request to determine how long a particular transaction is running. Execute the MONITOR SESSION and QUERY SESSION request to pinpoint the source of the problem. By running these two requests, you may not need to abort as many transactions as originally planned. For more information on QUERY SESSION, see *Teradata Vantage™ - Database Utilities*, B035-1102.

### **Relationship Between ABORT SESSION and MONITOR PHYSICAL RESOURCE or MONITOR VIRTUAL RESOURCE**

If you executed an ABORT SESSION request, data returned in a MONITOR PHYSICAL RESOURCE or MONITOR VIRTUAL RESOURCE request may be altered. Whether you notice the change in data depends on the scope of the ABORT SESSION request. For example, if you execute an ABORT SESSION and log off all of the sessions associated with a specific host (or client), the PEs associated with that client will report a large decrease in resource consumption. However, if the ABORT SESSION request only aborts one transaction from one session, you may not notice a change in AMP or PE resource use.

## IDENTIFY

Returns information on the locks blocking a session:

- Name of the user associated with a session
- User name of an object
- Database name of an object
- Name of a table

### Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .
<i>host_id</i>	SMALLINT	Logical ID of a host (or client). <i>host_id</i> cannot exceed 1023. A <i>host_id</i> of zero identifies the system console ID of the host. A combination of <i>host_id</i> and <i>session_no</i> identifies a user causing a block.
<i>session_no</i>	INTEGER	Session number. A combination of <i>host_id</i> and <i>session_no</i> identifies a user causing a block. To identify the user involved in a lock conflict, include the Blk_x_HostId and Blk_x_SessNo returned in a MONITOR SESSION response as input in the USING Data String for the IDENTIFY SESSION request.
<i>database_id</i>	INTEGER	ID of the database for this session. To identify the database name of the object involved in a block, include the Blk_x_ObjDBId returned in a MONITOR SESSION response as input in the USING Data String for the IDENTIFY DATABASE request.
<i>user_id</i>	INTEGER	ID of the user for this session. To identify the user involved in a lock conflict, include the Blk_x_UserId field returned in a MONITOR SESSION response as input in the USING Data String for the IDENTIFY USER request.
<i>table_id</i>	INTEGER	Unique ID of a table. To identify the table name of the object involved in a block, include the Blk_x_ObjTId returned in a MONITOR SESSION response as input in the USING Data String for the IDENTIFY TABLE request.

Element	Data Type	Description
		<b>Note:</b> The database name and user name are assigned an associated identifier when they are created. The IDENTIFY DATABASE or IDENTIFY USER request processes database and user names in the same manner because the database name and user name are almost equivalent.

**Note:**

Because the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserID fields returned by a MONITOR SESSION request may either be NULL or identify an internal session, the data returned by a MONITOR SESSION request that you use as input for the IDENTIFY request can result in error responses from IDENTIFY. If you use an internal session identifier as input to the IDENTIFY SESSION request, it will return an error message. The same error message is returned if you submit the IDENTIFY SESSION request for the DBQL/Teradata dynamic workload management software artificial internal session.

If you use a NULL UserID as input to the IDENTIFY DATABASE or IDENTIFY USER request, the system will return an error message.

**Monitor Privileges**

To use this request, you must have the MONSESSION privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

**Usage Notes - IDENTIFY**

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

The following table describes the different IDENTIFY options.

Option	Description
IDENTIFY DATABASE	Identify a locked database.
IDENTIFY SESSION	Identify a user (by session) who is causing a block. You can use IDENTIFY SESSION with a combination of <i>host_id</i> and <i>session_no</i> , or you can use IDENTIFY USER.
IDENTIFY TABLE	Identify a locked table.

Option	Description
IDENTIFY USER	Identify a user who is causing a block. You can use IDENTIFY SESSION , or you can use IDENTIFY USER <i>user_id</i> .

The following table lists the system table for each option. You can use Data Dictionary views to examine the information in each system table (see *Teradata Vantage™ - Data Dictionary*, B035-1092).

System Table Name	IDENTIFY Option
DBC.SessionTbl	IDENTIFY SESSION
DBC.DBase	IDENTIFY DATABASE IDENTIFY USER
DBC.TVM	IDENTIFY TABLE

## CLiv2 Response Parcels

Regardless of the form of the IDENTIFY request you execute, the response contains the following sequence of parcel types.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 2 ActivityCount = 1 ActivityType = 85 (PCLIDENTIFY)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the user name with the specified user ID or database ID, user name logged on as the specified session, or table name with the specified table ID.
EndStatement	11	6	StatementNo = 2-byte integer
EndRequest	12	4	None

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The Record returns the following field/column:

Field/Column Name	Data Type	Description
<i>Name</i>	VARCHAR(128) CHARACTER SET UNICODE NOT NULL	Name of the object (for example, database, user, or table) whose identifier was supplied by the IDENTIFY request.

## Sample Input - CLiv2 Request

The following example shows how the Request parcels for an IDENTIFY SESSION request, built by CLiv2, appear when sent to the database server using a *host\_id* of 348 and a *session\_no* of 1000.

### Note:

In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	20	Request	IDENTIFY SESSION
0003	Data	12	MonVerID	2
			HostId	348
			SessionNo	1000
0004	Resp	6	BufferSize	64000

The following example shows how the Request parcels for an IDENTIFY USER request, built by CLiv2, look when they are sent to the database server using a *user\_id* of 725.

### Note:

In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	21	Request	IDENTIFY DATABASE or IDENTIFY USER
0003	Data	10	MonVerID	2
			UserID	725
0004	Resp	6	BufferSize	64000

The next example shows how the Request parcels for an IDENTIFY TABLE request, built by CLIV2, look when they are sent to the database server using a *table\_id* of 183351.

**Note:**

In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	18	Request	IDENTIFY TABLE
0003	Data	10	MonVerID	2
			TableId	183351
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

With a *host\_id* of 52 and a *session\_no* of 31467, the IDENTIFY SESSION request might return the following values. These example values are returned in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
StatementNo: 1      ActivityCount: 1
ActivityType: 85    FieldCount: 1
DataInfo parcel:
FieldCount: 1
```

```
Record parcel.
  Parcel flavor:          10    Parcel body length:   31
  Name = "DBC
EndStatement.
EndRequest.
```

With a *database\_id* of 1012, the IDENTIFY DATABASE request might return the following values. These example values are returned in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
  StatementNo: 1    ActivityCount: 1
  ActivityType: 85    FieldCount: 1
DataInfo parcel:
  FieldCount: 1
Record parcel.
  Parcel flavor:          10    Parcel body length:   31
  Name = "weekly
EndStatement.
EndRequest.
```

With a *user\_id* of 1012, the IDENTIFY USER request might return the following values. These example values are returned in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
  StatementNo: 1    ActivityCount: 1
  ActivityType: 85    FieldCount: 1
DataInfo parcel:
  FieldCount: 1
Record parcel.
  Parcel flavor:          10    Parcel body length:   31
  Name = "weekly
EndStatement.
EndRequest.
```

With a *table\_id* of 63, the IDENTIFY TABLE request might return the following values. These example values are returned in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
  StatementNo: 1    ActivityCount: 1
  ActivityType: 85    FieldCount: 1
```

```

DataInfo parcel:
  FieldCount: 1
Record parcel.
  Parcel flavor:      10      Parcel body length:   31
  Name = "EventLog      ".
EndStatement.
EndRequest.

```

## Relationship Between IDENTIFY and MONITOR SESSION

PM/API can report on locks placed by any user or object with the MONITOR SESSION and IDENTIFY requests. The MONITOR SESSION request helps you identify the types of locks blocking a session.

The MONITOR SESSION request:

- Monitors the currently executing processes and reports blocks preventing sessions from doing useful work, for both application or utility locks causing the block.
- Tells you not only which session is blocked and on what type of object but also who is holding the lock. You can trace a blocked session back to the object locked and display the owner of the lock if your job is hung or is running very slowly and you suspect there is a lock conflict involved.
- On a query, reports the lock conflicts per session with the MoreBlocks field to indicate if there are more lock conflicts involved but not reported. The types of lock information returned for each session include:
  - User ID or user of host utility job causing a block
  - Type of object (for example, database, table, or row hash) causing a lock
  - Mode or severity of lock
  - Database identifier of object being locked
  - Table identifier of object being locked

By looking at the logical host ID of a session causing the block in combination with the session number of the session causing a block, you can uniquely identify the session that is causing a block.

Although you have the above lock information, you must use the IDENTIFY request to further identify the locks as either a user name, database name, or table name. Use the Blk\_ data values (or fields) returned in the MONITOR SESSION operation as input for an IDENTIFY request.

MONITOR SESSION, with IDENTIFY, is a more powerful diagnostic tool than the Show Locks utility accessed through Database Window (DBW), which displays information about only client utility locks on an object and does not report who the lock is blocking.

## Example: Write Lock Blocking Session

The following example output shows that SessionNo 1001 is blocked by SessionNo 1000 by a WRITE lock that is granted on a table object type. Records are sorted in HostId and then SessionNo order.

First Record:



```

HostId = 719
.
.
SessionNo = 1000
.
.
.

```

Second Record:

```

HostId = 719
.
.
SessionNo = 1001
.
.
Blk-1-HostId = 719
Blk-1-SessNo = 1000
Blk-1-UserID = 10
Blk-1-LMode = 'W'
Blk-1-OType = 'T'
Blk-1-ObjDBId = 12
Blk-1-ObjTId = 1097
Blk-1-Status = 'G'
Blk-2-HostId = NULL
.
.
.

```

**Note:**

When the Blk\_2\_ HostId, Blk\_2\_ SessNo, and Blk\_2\_ UserID values are returned as NULLs, this usually means that the blocking job is a HUT job that has logged off without releasing its lock.

Because the data returned here does not tell you which user is causing the block or which table is locked, you must next use the IDENTIFY request with the output from your MONITOR SESSION to return that information.

To identify the user causing the lock, execute the IDENTIFY request with:

```

HostId = 719
SessionNo = 1000

```

Or:

```
UserID = 10
```

To identify a locked database, execute the IDENTIFY request with:

```
ObjDBId = 12
```

To identify a locked table, execute the IDENTIFY request with:

```
ObjTId = 1097
```

## MONITOR AWT RESOURCE

Collects statistics on AMPs based on the in-use AMP Worker Tasks (AWTs).

### Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .
<i>Threshold 1</i>	SMALLINT NOT NULL	Minimum value for in-use AWTs to qualify an AMP for inclusion into this interval.
<i>Threshold 2</i>	SMALLINT	Start value for in-use AWTs to qualify an AMP for inclusion into this interval.
<i>Threshold 3</i>	SMALLINT	Start value for in-use AWTs to qualify an AMP for inclusion into this interval.
<i>Threshold 4</i>	SMALLINT	Start value for in-use AWTs to qualify an AMP for inclusion into this interval.
<i>Summary</i>	SMALLINT NOT NULL	0 = Return no record parcels in statement 2. 1 = Return the record parcels in statement 2, which are the data fields returned in Groups I through V only.
<i>Detail</i>	SMALLINT NOT NULL	0 = Return no detail information. 1 = Return AMP worker task usage information by each AMP. If Detail is enabled, a third statement is returned containing AMP worker task usage information by each AMP. If Detail is not enabled, a statement

Element	Data Type	Description
		is returned with no record parcels. For information on the record parcels returned, see <a href="#">Statement 3 - MONITOR AWT RESOURCE</a> .

**Note:**

You can define up to four thresholds for categorizing AMPs.

**Monitor Privileges**

To use this request, you must have the MONRESOURCE privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

**Usage Notes - MONITOR AWT RESOURCE**

This interface provides a snapshot of AMP utilization based on the number of AWTs in use within the MSGWORKNEW and MSGWORKONE message work types. These two message work types are a barometer for AMP activity/health. The MONITOR AWT RESOURCE request allows the specification of four thresholds which are used to categorize AMPs. AMPs are counted into categories defined by these thresholds and based on the AWT usage of the AMP.

This request also indicates the number of AMPs currently in some form of Flow Control. Flow Control can be defined at a node level and, therefore, may not have a system-wide consistent definition. This indication is applicable only to those messages to which Flow Control is applicable.

MONITOR AWT RESOURCE provides information on the AMPs having the highest and lowest in-use counts within the system. When duplicate in-use counts exist, the AMP information returned is for the AMPs with the largest VProc ID.

Thresholds must be defined in ascending order and cannot contain gaps. If a gap is detected, an error message is returned.

Thresholds that are not used must be set to -1 (or NOT NULL). The minimum valid threshold value is zero.

**CLlv2 Response Parcels**

The MONITOR AWT RESOURCE request is treated internally as a one statement request that generates two responses. The statement response returned from the database contains the following sequence of parcel types.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments and Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = 1 ActivityType = 175 (PCLMONAWTRESSTMT)
DataInfo	71	6 to 64100	Optional: This parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul>	Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo-1. For an example of this record, see <a href="#">Statement 1</a> .
EndStatement	11	6	StatementNo = 2-byte integer=1
Success	8	18 to 273	StatementNo = 2 ActivityCount = 1 ActivityType = 175 (PCLMONAWTRESSTMT)
DataInfo	71	6 to 64100	Optional: This parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>• 5 to 64100(record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul>	Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo-2. For an example of this record, see <a href="#">Statement 2</a> .
EndStatement	11	6	StatementNo = 2-byte integer=1
EndRequest	12	4	None
Success	8	18 to 273	StatementNo = 3 ActivityCount = 1 ActivityType = 175 (PCLMONAWTRESSTMT)
DataInfo	71	6 to 64100	Optional: This parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul>	Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo-3. For an example of this record, see <a href="#">Statement 3 - MONITOR AWT RESOURCE</a> .
EndStatement	11	6	StatementNo = 3-byte integer=1
EndRequest	12	4	None

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The MONITOR AWT RESOURCE request returns the following information: the number of AWTs in use within the MSGWORKNEW and MSGWORKONE message work types; the number of AMPs currently in some form of Flow Control; and the number of AMPs having the highest and lowest in-use counts within the system.

The output returns three statements:

- The first statement provides information about the collection rate.
- The second statement provides information about AMPs and the AMPs AWT distribution across the defined thresholds.
- The third statement provides AMP worker task usage information by each AMP.

### Statement 1

The first statement is a Record parcel format containing:

Field/Column Name	Data Type	Description
SampleSec	INTEGER NOT NULL	Duration of the collection period, in seconds. This field contains the Monitor resource collection rate (ResMonitor).
CollectionDate	DATE NOT NULL	Date the Monitor AWT cache was last refreshed.
CollectionTime	FLOAT NOT NULL	Time the Monitor AWT cache was last refreshed.

To avoid excessive AWT data collection, Monitor AWT Resource checks if the data in the buffer is still valid in relation to the value set for SampleSec (ResMonitor rate) and only collects new data if it has expired.

### Statement 2

The second statement is a Record parcel format containing information about AMPs and the AMPs AWT distribution across the defined thresholds.

Field/ Column Name	Data Type	Description
Interval 1 Count	INTEGER	Number of AMPs in the system whose in-use AWT counts fall at, or above, the Threshold 1 value and do not qualify for the higher thresholds.
Interval 2 Count	INTEGER	Number of AMPs in the system whose in-use AWT counts fall at, or above, the Threshold 2 value and do not qualify for the higher thresholds.
Interval 3 Count	INTEGER	Number of AMPs in the system whose in-use AWT counts fall at, or above, the Threshold 3 value and do not qualify for the higher thresholds.
Interval 4 Count	INTEGER	Number of AMPs in the system whose in-use AWT counts fall at, or above, the Threshold 4 value and do not qualify for the higher thresholds.
Flow Control	INTEGER	Number of AMPs currently in some form of Flow Control.
High AMP 1 VprocId	INTEGER	Vproc ID of the AMP with the highest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
High AMP 1 In-Use Count	INTEGER	In-use count associated with the AMP 1 Vproc ID. This is only applicable if AMP 1 is defined.
High AMP 2 VprocId	INTEGER	Vproc ID of the AMP with the next highest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
High AMP 2 In-Use Count	INTEGER	In-use count associated with the AMP 2 Vproc ID. This is only applicable if AMP 2 is defined.
High AMP 3 VprocId	INTEGER	Vproc ID of the AMP with the next highest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
High AMP 3 In-Use Count	INTEGER	In-use count associated with the AMP 3 Vproc ID. This is only applicable if AMP 3 is defined.
High AMP 4 VprocId	INTEGER	Vproc ID of the AMP with the next highest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
High AMP 4 In-Use Count	INTEGER	In-use count associated with the AMP 4 Vproc ID. This is only applicable if AMP 4 is defined.
High AMP 5 VprocId	INTEGER	Vproc ID of the AMP with the next highest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
High AMP 5 In-Use Count	INTEGER	In-use count associated with the AMP 5 Vproc ID. This is only applicable if AMP 5 is defined.
Low AMP 1 VprocId	INTEGER	Vproc ID of the AMP with the lowest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
Low AMP 1 In-Use Count	INTEGER	In-use count associated with the AMP 1 Vproc ID. This is only applicable if AMP 1 is defined.

Field/ Column Name	Data Type	Description
Low AMP 2 VprocId	INTEGER	Vproc ID of the AMP with the lowest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
Low AMP 2 In-Use Count	INTEGER	In-use count associated with the AMP 2 Vproc ID. This is only applicable if AMP 2 is defined.
Low AMP 3 VprocId	INTEGER	Vproc ID of the AMP with the lowest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
Low AMP 3 In-Use Count	INTEGER	In-use count associated with the AMP 3 Vproc ID. This is only applicable if AMP 3 is defined.
Low AMP 4 VprocId	INTEGER	Vproc ID of the AMP with the lowest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
Low AMP 4 In-Use Count	INTEGER	In-use count associated with the AMP 4 Vproc ID. This is only applicable if AMP 4 is defined.
Low AMP 5 VprocId	INTEGER	Vproc ID of the AMP with the lowest in-use count in the system. A value of -1 (or NULL) indicates this field is not defined.
Low AMP 5 In-Use Count	INTEGER	In-use count associated with the AMP 5 Vproc ID. This is only applicable if AMP 5 is defined.
Flow Control 1 VprocId, Flow Control 2 VprocId, Flow Control 3 VprocId, Flow Control 4 VprocId, Flow Control 5 VprocId	INTEGER	Vproc ID of the AMP in flow control. A value of -1 (or NULL) indicates this field is not defined.

### Statement 3 - MONITOR AWT RESOURCE

The third statement is a Record parcel format containing AMP worker task usage information by each AMP.

Field/Column Name	Data Type	Description
VprocNo	SMALLINT NOT NULL	AMP vproc number.
AvailableAWTs	SMALLINT NOT NULL	Number of available AMP worker tasks in each AMP.
InUseAWTs	SMALLINT NOT NULL	Number of active AMP worker tasks in each AMP.
MsgCount	INTEGER	Number of messages currently queued for delivery to each AMP.

Field/Column Name	Data Type	Description
	NOT NULL	
DQMsgCount	INTEGER NOT NULL	Number of messages processed by each AMP.
AvailableAWTsForAll	SMALLINT	The number of available AMP worker tasks in the unreserved pool in each AMP.

### Sample Input - CLv2 Request

The following example shows how the parcels for a MONITOR AWT RESOURCE request, built by CLv2, appear when sent to the database server.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size. However, the minimum size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	16	Request	MONITOR AWT RESOURCE
0003	Data	57	MonVerId Threshold 1 Threshold 2 Threshold 3 Threshold 4 Summary Detail	6 10 12 14 15 1 0
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

The MONITOR AWT RESOURCE request returns values approximately as shown below when TASM Workloads are enabled and the following input data is specified:



```

Threshold 1 = 10
Threshold 2 = 12
Threshold 3 = 14
Threshold 4 = 15
    Summary = 1
    Detail = 1

```

For information on TASM rules, see *Teradata® Viewpoint User Guide*, B035-2206.

The MONITOR AWT RESOURCE request commonly returns values in text character format. Your application program may return the values in a different format or display.

---

**Note:**

You can rename the SampleSec field in your application. In the output below, the SampleRate value is the SampleSec value.

---

Pay attention to SampleRate when interpreting the results of this request.

```

SampleRate: 30
Collection Date/Time: 06/15/2011 18:32:44.00
SUCCESS parcel:
StatementNo=2, ActivityCount=1,
ActivityType=175, FieldCount=30
Intervals:
          1      2      3      4
          4      0      0      0
Flow Control = 0
*** HIGH AMP ***
          1      2      3      4      5
VprocId:   3      2      1      0     -1
InUseCount: 1      1      1      1      0
*** LOW AMP ***
          1      2      3      4      5
VprocId:   3      2      1      0     -1
InUseCount: 1      1      1      1      0
*** FLOW CONTROL INFO ***
          1      2      3      4      5
VprocId   -1     -1     -1     -1     -1
SUCCESS parcel:
StatementNo=3, ActivityCount=4,
ActivityType=175, FieldCount=5
VProcNo  AvailableAWTs  InUseAWTs  MsgCount  DQMsgCount
-----  -

```

0	49	2	0	111135
1	49	1	0	124561
2	49	1	0	156017
3	49	1	0	122913

## MONITOR PHYSICAL CONFIG

Collects overall information on node availability. Node status information is returned for all nodes in the system.

### Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .

### Monitor Privileges

To use this request, you must have any of the following monitor privileges as part of your default role or any of these privileges must be granted directly to you:

- ABORTSESSION
- MONRESOURCE
- MONSESSION
- SETRESRATE
- SETSESSRATE

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR PHYSICAL CONFIG

MONITOR PHYSICAL CONFIG is most useful when used with the MONITOR PHYSICAL SUMMARY request for doing a quick overall system health check. For more information, see [Relationship Between MONITOR PHYSICAL CONFIG and MONITOR PHYSICAL SUMMARY](#).

You can use the MONITOR PHYSICAL CONFIG request instead of dumping the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view) to check for a physical problem with the system.

## CLiv2 Response Parcels

The MONITOR PHYSICAL CONFIG request is treated internally as a two statement request, with each statement generating a response. The database returns the two statement response containing the following sequence of parcels.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = 1 ActivityType = 92 (PCLMONPCONFIG)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the BYNET status data and the type of system running the Teradata Vantage software.
EndStatement	11	6	StatementNo = 2-byte integer
Success	8	18 to 273	StatementNo = 2 ActivityCount = Number of nodes ActivityType = 92 (PCLMONPCONFIG)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. Multiple record parcels are returned that consist of a record for each node in the system. This record contains node-specific information; one record per node.
EndStatement	11	6	StatementNo = 2-byte integer
EndRequest	12	4	None

For descriptions of the flavor field and length field, see *Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems*, B035-2417 or *Teradata® Call-Level Interface Version 2 Reference for*

*Workstation-Attached Systems*, B035-2418. Within the parcel body fields, the order of items and their data types and lengths are determined by the USING Phrase.

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Statement 1

The Record parcel in the first statement of the MONITOR PHYSICAL CONFIG response returns the following values:

Column	Field/Column Name	Data Type	Description
1	NetAUUp	VARCHAR(1) NOT NULL	<p>Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:</p> <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <p><b>Note:</b> This output parameter is available on monitor software version 9 or later only.</p>
2	NetBUUp	VARCHAR(1) NOT NULL	<p>Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:</p> <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <p><b>Note:</b> This output parameter is available on monitor software version 9 or later only.</p>
3	SystemType	VARCHAR(7) NOT NULL	<p>Type of system running the Teradata Vantage software, such as 5650, 6700, or 'Other'.</p> <p>If all the nodes in the system are the same type, this field returns the type of the system.</p> <p>If any of the nodes are of a different type, this field returns 'Mixed'.</p>

Column	Field/Column Name	Data Type	Description
			<b>Note:</b> This output parameter is available on monitor software version 9 or later only.
4	SystemName	VARCHAR(15)	Name of the system running Teradata.  <b>Note:</b> If you are using MONITOR software version ID ( <i>mon_ver_id</i> ) 10 or earlier, this output parameter is not available.

## Statement 2

The response to the second statement results in multiple Record parcels that consist of a record for each node in the system. For example, if you have two nodes, two records are returned with specific information for each processor.

Records are sorted based on NodeID. The following table shows the order in which the Record parcel returns the data.

Field/Column Name	Data Type	Description
ProclD	INTEGER NOT NULL	ID associated with a node. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.
Status	VARCHAR(1) NOT NULL	Status of the node associated with this record: <ul style="list-style-type: none"> <li>• U = Up/online</li> <li>• D = Down/offline</li> <li>• S = Standby</li> </ul> A node is up (U) when it is: <ul style="list-style-type: none"> <li>• Configured into the system</li> <li>• Online</li> <li>• Capable of actively performing tasks associated with normal database activity</li> </ul> Down (D) represents all other potential states. Standby (S) indicates the node is ready to join the configuration in place if another node goes down. When the node status is Standby, the SystemType, NetAUp, and NetBUp fields are not available and NULL or spaces will be returned.
CPUType	VARCHAR (7) NOT NULL	Type of central processing unit (CPU) installed in this node , for example, 'Pentium', 'PentPro', or 'Unknown'.
CPUCount	INTEGER	Number of CPUs in this node.

Field/Column Name	Data Type	Description
	NOT NULL	
SystemType	VARCHAR (7)	Type of system running the Teradata Vantage software, such as 5650, 6700, or 'Other'. <b>Note:</b> This output parameter is available on monitor software version 9 or later only.
CliqueNo	SMALLINT NOT NULL	Clique number of the node. <b>Note:</b> This output parameter is available on monitor software version 9 or later only.
NetAUp	VARCHAR(1)	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = Node BYNET is up/online.</li> <li>• D = Node BYNET is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <b>Note:</b> This output parameter is available on monitor software version 9 or later only.
NetBUp	VARCHAR(1)	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = Node BYNET is up/online.</li> <li>• D = Node BYNET is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <b>Note:</b> This output parameter is available on monitor software version 9 or later only.
PhyMemory	INTEGER	Size of the physical memory of the node in MBs. <b>Note:</b> If you are using MONITOR software version ID ( <i>mon_ver_id</i> ) 10 or earlier, this output parameter is not available.

### Sample Input - CLIV2 Request

The following example shows how the parcels for a MONITOR PHYSICAL CONFIG request, built by CLIV2, look when sent to the database server.

**Note:**

In the following example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	27	Request	MONITOR PHYSICAL CONFIG
0003	Data	6	MonVerID	9
0004	Resp	6	BufferSize	64000

**Sample Input - Teradata JDBC Driver Request**

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

**Sample Output**

Using monitor software version ID 11, this request might return the following values in character text format.

**Note:**

Your application program may display returned values in a different format.

```
Submitting request MONITOR PHYSICAL CONFIG; ...
```

```
NetAUp:  U NetBUp:  U
SystemType: 5500C
SystemName: localhost
```

```
1 node(s) found
```

```
ProcId:    10001 (1-1) Status: [ U]   CPUType: [Xeon   ]   CPUCount: 1
SystemType: [5500C  ]
CliqueNo: 0
NetAUp: [  ] NetBUp: [  ]
PhyMemory: 5720
```

## Relationship Between MONITOR PHYSICAL CONFIG and MONITOR PHYSICAL SUMMARY

Use the MONITOR PHYSICAL SUMMARY request with the MONITOR PHYSICAL CONFIG request for an overall system status. These are low overhead requests.

- Execute the MONITOR PHYSICAL SUMMARY request every 5 or 10 minutes for a low-cost, continuous monitoring of your system.
- Execute the MONITOR PHYSICAL CONFIG request to get a picture of your system configuration at defined times, such as at the beginning of a day, various times during the day, or when the system is down.

Use these requests to spot problems, such as abnormal Central Processing Unit (CPU) load balancing, and possible sources of system performance bottlenecks. For example, if the High/LowCPUUse figures are consistently widely separated and do not approximate the AvgCPU figure, you may need to evaluate whether the system is using available resources efficiently. How often you check your system depends on the size of your system and the type of applications your system runs.

Knowledge of the overall system status can help you to determine these three concerns.

Concern	Comments
When to run production applications, especially large ones	For example, if you have a down node, some Access Module Processors (AMPs) and Parsing Engines (PEs) may migrate to other nodes. It may be less costly to recover the node first and run the job than to run the job without full system availability.
Why an application runs more slowly than usual	This situation may be caused by a down node, which causes the online nodes to run more than the optimal number of AMPs and PEs. This, in turn, could cause your application to run more slowly.
Whether all nodes have come back up after a system restart	Examine the Status value returned in a MONITOR PHYSICAL CONFIG request to determine whether each node is up or down.

If the data returned from a MONITOR PHYSICAL SUMMARY query does not give you enough information (for example, you need BYNET or CPU% busy information), use the MONITOR PHYSICAL RESOURCE request to obtain more detailed resource usage data.

The data returned by the MONITOR PHYSICAL CONFIG request is an abbreviated form of the data returned by the MONITOR PHYSICAL RESOURCE request.



**Note:**

The MONITOR PHYSICAL CONFIG and MONITOR PHYSICAL SUMMARY requests do not return the status of non-database nodes. However, the MONITOR PHYSICAL RESOURCE and MONITOR PHYSICAL SUMMARY requests accumulate and return data on some resources consumed by non-Teradata applications running on database nodes. To determine the resources consumed by non-Teradata applications, compare:

- Data returned by the MONITOR PHYSICAL RESOURCE request with that of the MONITOR VIRTUAL RESOURCE request (that is, the subtotal of the various resource statistics by node).
- Data returned by the MONITOR PHYSICAL SUMMARY request with that of the MONITOR VIRTUAL SUMMARY request (that is, the subtotal/average by node of the various resource statistics in the MONITOR VIRTUAL SUMMARY).

## MONITOR PHYSICAL RESOURCE

Collects RSS data and returns node-specific data.

### Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later only. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .

### Monitor Privileges

To use this request, you must have the MONRESOURCE privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR PHYSICAL RESOURCE

Because information is given on the detailed resource usage of each node, performance concerns can be isolated by node.

Use the MONITOR PHYSICAL RESOURCE request to:

- Expand on the data reported by the MONITOR PHYSICAL SUMMARY request.  
In your initial problem analysis, a MONITOR PHYSICAL SUMMARY request may indicate a performance or system problem. MONITOR PHYSICAL RESOURCE allows you to collect RSS data on a node by node basis.

- Continually monitor your system.

Monitor your system on a periodic basis, for example, every 10 minutes. Use this request to build a normal baseline profile for your system. When you notice something abnormal or a user complains that a job is slow (such as the last reading is significantly different from the normal baseline reading), this request can tell you:

- If there is a parallel efficiency problem
- A constraint to throughput
- And which node is causing it.

- Determine whether a new application can be added to the current system load without disruption.

The node usage information collected by this request can help you evaluate the impact of adding new applications to an already heavily utilized system and help you plan potential system upgrades.

- Help resolve problems that session-level usage information cannot resolve.

When the MONITOR SESSION request does not show any cause for the problem, the MONITOR PHYSICAL RESOURCE request provides information on congestion, memory allocations, BYNET outages, and system status.

The MONITOR PHYSICAL RESOURCE request provides the following information:

- How the system is being used (for example, the percentage of CPU usage by node)
- How system resource usage is spread across the nodes
- How much physical disk Input/Output (I/O), BYNET traffic, or host reads and writes are occurring
- Whether congestion or excessive swapping is a problem on any node or group of nodes

The MONITOR PHYSICAL RESOURCE request returns some of the same fields found in the resource usage tables. You can use both MONITOR PHYSICAL RESOURCE and resource usage data for problem detection. Unlike resource usage data, MONITOR PHYSICAL RESOURCE data is near real time, requires less overhead to produce, but is less comprehensive. MONITOR PHYSICAL RESOURCE data helps detect:

- Poor Node CPU parallel efficiency
- Poor disk parallel efficiency
- A higher than expected disk read/write ratio
- A high swap I/O rate

If the MONITOR PHYSICAL RESOURCE request does not provide enough detailed data for problem detection, run one or more of the resource usage macros. For more information, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

For some of the MONITOR PHYSICAL RESOURCE and MonitorPhysicalResource fields, NULL returns if:

- A node is down or offline
- The ResMonitor is set to zero

---

**Note:**

You must set the ResMonitor rate to a nonzero value to allow the MONITOR PHYSICAL RESOURCE request or MonitorPhysicalResource function to return meaningful data.

---

- A request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate

After a system outage or a change in the ResMonitor rate, do not request data again until after completion of the first collection period requested after the crash or change in rate. Otherwise, the data returned will contain NULL except NetAUp, NetBUUp, SampleSec, ProclId, AMPCount, PECCount, and Status, and may not be fully representative. The in-memory counters reset after a crash. Typically the contents of the counters are not well defined until a full collection period has elapsed. If you were logged on prior to the system outage, and you issue the first MONITOR PHYSICAL RESOURCE request or MonitorPhysicalResource function after the outage, you will receive a warning that the database system has been restarted.

## CLv2 Response Parcels

The MONITOR PHYSICAL RESOURCE request is treated internally as a two statement request with each statement generating a response. The database returns a two statement response containing the following sequence of parcel types.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = 1 ActivityType = 96 (PCLMONPRES)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. One record is returned. It contains the duration of the collection period (in seconds), BYNET status, and the data and time of when the Physical Resource cache was last refreshed.
EndStatement	11	6	StatementNo = 2-byte integer

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 2 ActivityCount = Number of nodes ActivityType = 96 (PCLMONITOPRES)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicReq parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. One record per node is returned. Each record contains a description for each node, including BYNET status.
EndStatement	11	6	StatementNo = 2-byte integer
EndRequest	12	4	None

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Statement 1

The response to the first statement results in a Record parcel containing global data about the collection duration and BYNET status that is generated once for the whole system. The following table shows the order in which the data is returned.

Field/Column Name	Data Type	Description
NetAUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>U = All node BYNETs are up/online.</li> <li>D = One or more node BYNETs is down/offline.</li> <li>"" = A temporary condition where the BYNET data is not available.</li> </ul>
NetBUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>U = All node BYNETs are up/online.</li> <li>D = One or more node BYNETs is down/offline.</li> <li>"" = A temporary condition where the BYNET data is not available.</li> </ul>

Field/Column Name	Data Type	Description
SampleSec	SMALLINT NOT NULL	Duration of the collection period in seconds. This field is equivalent to the ResMonitor rate. See <a href="#">Data Collection</a> and <a href="#">SET RESOURCE RATE</a> for more information on ResMonitor.
CollectionDate	DATE NOT NULL	Date the Physical Resource cache was last refreshed.
CollectionTime	FLOAT, NOT NULL	Time the Physical Resource cache was last refreshed.

## Statement 2

The response to the second statement results in multiple Record parcels that consist of a one record of 32 fields for each node in the system. For example, if you have 50 nodes, 50 records are returned with specific information for each node. One record describes the collection period and BYNET status for the entire system.

The following table shows the order in which the Record parcel returns the data.

Field/Column Name	Data Type	Description
Procid	INTEGER NOT NULL	ID associated with a node. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.
AmpCount	SMALLINT NOT NULL	Current number of AMPs currently executing on the associated node.
PECount	SMALLINT NOT NULL	Current number of active PEs on the associated node.
CPUUse	FLOAT range 0 - 100%	% of CPU usage not spent being idle. This value is computed from ResUsageSpma table data as: PercntUser + PercntService This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
PercntIOWait	FLOAT	% of CPU resources in idle and waiting for I/O completion. This value is computed from ResUsageSpma data as follows, where x is the number of CPUs: $\text{CPU IOWAIT} / (x * \text{SampleSec})$ This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
PercntService	FLOAT	% of CPU resources spent in PDE user service processing. The value is computed from the ResUsageSpma table data, where x represents the number of CPUs:

Field/Column Name	Data Type	Description
		<p><math>CPUUServ / (x * SampleSec)</math></p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
PercntUser	FLOAT	<p>% of CPU resources spent in non-service user code processing. This value is computed from the ResUsageSpma table data, where x represents the number of CPUs:</p> <p><math>CPUUExec / (x * SampleSec)</math></p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
Status	VARCHAR (1) NOT NULL	<p>Status of the node associated with this record:</p> <ul style="list-style-type: none"> <li>• U = Up/online</li> <li>• D = Down/offline</li> <li>• S = Standby</li> </ul> <p>A node is up (U) when it is:</p> <ul style="list-style-type: none"> <li>• Configured into the system</li> <li>• Online</li> <li>• Capable of actively performing tasks associated with normal database activity</li> </ul> <p>Down (D) represents all other potential states.</p> <p>Standby (S) indicates the node is ready to join the configuration in place if another node goes down. When the node status is Standby, the SystemType, NetAUp, and NetBUp fields are not available and NULL or spaces will be returned.</p>
NetAUse	FLOAT	<p>% of BYNET A usage. This is the actual BYNET receiver usage. (The BYNET transmitter usage is maintained in resource usage separately and is typically lower than the receiver usage. This is caused by multicasts, where one transmitter sends a message to many receivers.) This value is computed from the ResUsageSpma table data as:</p> <p><math>((NetSamples - NetTxIdle) / NetSamples) * 100.0</math></p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
DiskUse	FLOAT	<p>% of disk usage per node.</p> <p>This value is computed from ResUsageSldv table data as follows, assuming n is the number of ldv devices used by this node:</p> <p><math>(LdvOutReqTime\ 1 + \dots + LdvOutReqTime\ n) / (n * SampleSec)</math></p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p> <p><b>Note:</b></p> <p>The DiskUse field does not take into account overlapping of operations among multiple storage devices, but it allows for the possibility of multiple requests for the same device.</p>

Field/Column Name	Data Type	Description
DiskReads	FLOAT	<p>Total number of physical disk reads per node during the collection period. This value is computed from ResUsageSldv table data as follows, assuming <math>n</math> is the number of ldv devices used by this node:</p> $\text{LdvReads } 1 + \dots + \text{LdvReads } n$ <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
DiskWrites	FLOAT	<p>Total number of physical disk writes per node during the collection period. This value is computed from ResUsageSldv table data as follows, assuming <math>n</math> is the number of ldv devices used by this node:</p> $\text{LdvWrites } 1 + \dots + \text{LdvWrites } n$ <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
DiskOutReqAvg	FLOAT	<p>Average number of outstanding disk requests per disks (averaged over all the disks on a node). This value can be used to monitor the load on the disks and indicate problems with the throughput if the level becomes too high.</p> <p>This value is computed from ResUsageSldv table data as follows, assuming <math>n</math> is the number of ldv devices used by this node:</p> $((\text{LdvOutReqSum } 1 / \text{NULLIFZERO}(\text{LdvOutReqDiv } 1)) + \dots + (\text{LdvOutReqSum } n / \text{NULLIFZERO}(\text{LdvOutReqDiv } n))) / n$ <p>The range of the value is typically 0 to 25.</p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
HostBlockReads	FLOAT	<p>Number of message blocks (one or more messages sent in one physical group) received from all clients. This value is computed from ResUsageShst data, assuming <math>n</math> is the number of host channel and network connections on this node:</p> $\text{HostBlockReads } 1 + \dots + \text{HostBlockReads } n$ <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
HostBlockWrites	FLOAT	<p>Number of message blocks (that is, one or more messages sent in one physical group) sent to all hosts. For node displays, this value is computed from ResUsageShst data, assuming <math>n</math> is the number of host channel and network connections on this node:</p> $\text{HostBlockWrites } 1 + \dots + \text{HostBlockWrites } n$ <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>
SwapReads	FLOAT	<p>Number of pages/segments read into node memory from the disk during the collection period after a prior write/drop. This value is computed from the ResUsageSpma table data as MemCtxtPageReads.</p> <p>For information on MemCtxtPageReads, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a>.</p>

Field/Column Name	Data Type	Description
SwapWrites	FLOAT	Number of pages/segments written to swap area from node memory during the collection period. This value is computed from the ResUsageSpma table data as MemCtxtPageWrites. For information on MemCtxtPageReads, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i> , B035-1099. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
SwapDrops	FLOAT	Number of pages/segments dropped from node memory during the collection period due to swapping. This field returns zero.
MemAllocates	FLOAT	<b>Note:</b> This field is obsolete and returns zero or NULL.
MemAllocateKB	FLOAT	Value represents the change in the node-level memory. MemAllocateKB represents a delta from the previous reporting period. It reports negative values as less memory is used. This value is calculated from the ResUsageSpma column: MemVprAllocKB This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
MemFailures	FLOAT	<b>Note:</b> This field is obsolete and returns zero or NULL.
MemAgings	FLOAT	<b>Note:</b> This field is obsolete and returns zero or NULL.
NetReads	FLOAT	Number of Reads from the BYNET to the node. This value is computed from the ResUsageSpma table data as: NetRxCircBrd + NetRxCircPtP This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
NetWrites	FLOAT	Number of messages written from the node to the BYNET during the collection period. For node-level displays, the value is computed from the ResUsageSpma table data as: NetTxCircBrd + NetTxCircPtP This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL RESOURCE</a> .
NetAUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = Node BYNET is up/online.</li> <li>• D = Node BYNET is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul>



Field/Column Name	Data Type	Description
		<b>Note:</b> This output parameter is available on monitor software version 9 or later only.
NetBUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = Node BYNET is up/online.</li> <li>• D = Node BYNET is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <b>Note:</b> This output parameter is available on monitor software version 9 or later only.

### Sample Input - CLiv2 Request

The following example shows how the parcels, built by CLiv2, for a MONITOR PHYSICAL RESOURCE request look when sent to the database.

#### Note:

In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	20	Request	MONITOR PHYSICAL RESOURCE
0003	Data	6	MonVerID	9
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This request might return the following values in character text format (a record for each node). Your application program may display returned values in a different format.

**Note:**

You can rename the SampleSec field in your application. In the output below, the SampleRate value is the SampleSec value.

Pay attention to SampleRate when interpreting the results of this request. Use monitor version 13 when running the following example:

```
Submitting request MONITOR PHYSICAL RESOURCE; ...

NetAUp:  U NetBUp:  U
SampleRate: 60
Collection Date/Time: 07/06/2016 13:48:00.00

ProcId:    10001 (1-1) AmpCount: 4   PEGCount: 2

CPUUse:  100.0   PrcntKernel:  0.0   PrcntService:  2.2   PrcntUser: 97.8

Status: U

NetAUse:      0.0   DiskUse:      10.7
DiskReads:    36.00   DiskWrites: 6500.00   DiskOutReqAvg:  0.19

HstBlkRds:    8.00   HstBlkWrts:  8.00

SwapReads:    0.00   SwapWrites:  0.00   SwapDrops:    0.00

MemAllocates:      0.00   MemAllocateKB: 16.00   MemFailures:      0.00
MemAgings:      0.00

NetReads:      0.00   NetWrites:    0.00

NetAUp:  U NetBUp:  U
```

**Relationship Between MONITOR PHYSICAL RESOURCE and ABORT SESSION**

If you executed an ABORT SESSION request, data returned in a MONITOR PHYSICAL RESOURCE or MONITOR VIRTUAL RESOURCE request may be altered. Whether you notice the change in data depends on the scope of the ABORT SESSION request. For example, if you execute an ABORT SESSION and log off all of the sessions associated with a specific host (or client), the PEs associated with that client will report a large decrease in resource consumption. However, if the ABORT SESSION request only aborts one transaction from one session, you may not notice a change in AMP or PE resource use.

## Relationship Between MONITOR PHYSICAL RESOURCE and SET RESOURCE RATE

You must execute the SET RESOURCE RATE request to activate resource data collection before you execute a MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request. This means that you must set the resource monitoring rate (ResMonitor) to nonzero. If the ResMonitor rate is set to zero, you will receive an error message.

A change in the resource collection rate by User A, for example, may affect the data reported by MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request made by User B. If the ResMonitor rate is altered, User B receives a warning message when executing a subsequent MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request.

## MONITOR PHYSICAL SUMMARY

Collects global summary information that includes the following types of information:

- CPU usage (average, high, and low)
- Disk usage (average, high, and low)
- BYNET usage (total, up/down)
- Rate information (resource logging rate and resource monitoring rate)
- Current software release and version numbers

### Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT, NOT NULL	MONITOR software version ID. This can be version 2 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .

### Monitor Privileges

To use this request, you must have any one of the following monitor privileges as part of your default role or any of these privileges must be granted directly to you:

- ABORTSESSION

- MONRESOURCE
- MONSESSION
- SETRESRATE
- SETSESSRATE

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR PHYSICAL SUMMARY

For some of the MONITOR PHYSICAL SUMMARY and MonitorPhysicalSummary fields, NULL returns if:

- A node is down or offline
- The ResMonitor is set to zero

---

### Note:

You must set the ResMonitor rate to a nonzero value to allow the MONITOR PHYSICAL SUMMARY request or MonitorPhysicalSummary function to return meaningful data.

---

- A request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate

After a system outage or a change in the ResMonitor rate, do not request data again until after completion of the first collection period requested after the crash or change in rate. Otherwise, the data returned will contain NULL except NetAUp, NetBUp, SampleSec, ProclD, AMPCount, PECCount, and Status, and may not be fully representative. The in-memory counters reset after a crash. Typically the contents of the counters are not well-defined until a full collection period has elapsed. If you were logged on prior to the system outage, and you issue the first MONITOR PHYSICAL SUMMARY request or MonitorPhysicalSummary function after the outage, you will receive a warning that the database system has been restarted.

If PE only (AMP-less) nodes exist and if the nodes should be included in the MONITOR PHYSICAL SUMMARY statistics calculation, you must set the DBS Control field, MPS\_IncludePEOnlyNodes, to TRUE.

---

### Note:

In normal use of Monitor Physical Summary, the MPS\_IncludePEOnlyNodes field is set to FALSE.

---

By default, the statistics for PE only nodes are excluded in the calculation of MONITOR PHYSICAL SUMMARY statistics. For information about the MPS\_IncludePEOnlyNodes field, see *Teradata Vantage™ - Database Utilities*, B035-1102.

## CLv2 Response Parcels

The response returned from the database resembles a summary of the type of response returned by a MONITOR PHYSICAL SUMMARY request. The response is one row of 22 fields. The response returned from the database contains the following sequence of parcel types.

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = 1 ActivityType = 94 (PCLMONPSUMMARY)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the Data or IndicData physical summary information and the date and time the Physical Resource cache was last refreshed.
EndStatement	11	6	StatementNo = 2-byte integer
EndRequest	12	4	None

For more information on parcel body fields, see the appropriate section in *Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems*, B035-2417 or *Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems*, B035-2418.

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The following table describes the resource usage information returned from the Record parcel for the MONITOR PHYSICAL SUMMARY response.

Field/Column Name	Data Type	Description
AvgCPU	FLOAT	Average % CPU usage (CPUUse) time of all online nodes currently in the Vantage configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .

Field/ Column Name	Data Type	Description
AvgDisk	FLOAT	Average % disk usage (from DiskUse) of all online nodes currently in the Vantage configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
AvgDiskIO	FLOAT	Average number DiskReads and DiskWrites for all online nodes currently in the Vantage configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
HighCPUUse	FLOAT	Highest CPUUse number associated with any online node that is currently part of the Vantage configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
HighCPUProcId	INTEGER	ID of a node with CPUUse equal to the value reported as HighCPUUse. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowCPUUse	FLOAT	Lowest CPUUse number associated with any online node that is currently part of the database configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowCPUProcId	INTEGER	ID of a node with CPUUse equal to the value reported as LowCPUUse. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
HighDisk	FLOAT	Highest % disk usage (from DiskUse) associated with any online node that is currently part of the database configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
HighDiskProcId	INTEGER	ID of a node with DiskUse equal to the value reported as HighDisk. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowDisk	FLOAT	Lowest % disk usage (from DiskUse) associated with any online node that is currently part of the database configuration.

Field/ Column Name	Data Type	Description
		This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowDiskProcid	INTEGER	ID of a node with DiskUse equal to the value reported as LowDisk. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL when LowDisk is NULL.
HighDiskIO	FLOAT	Highest DiskReads and DiskWrites number associated with any online node that is currently active in the database configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
HighDiskIOProcid	INTEGER	ID of a node with DiskReads and DiskWrites equal to the value reported as HighDiskIO. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowDiskIO	FLOAT	Lowest DiskReads and DiskWrites number associated with any online node that is currently part of the database configuration. This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .
LowDiskIOProcid	INTEGER	ID of a node with DiskReads and DiskWrites equal to the value reported as LowDiskIO. This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123. This value is NULL when LowDiskIO is NULL.
NetUse	FLOAT	% of total BYNET use (that is, average of the online BYNETs). If both BYNETs are up, the value is computed from ResUsageSpma table data as: $NetUse = Average\ NetAUse\ per\ node / NetCount$ where: <ul style="list-style-type: none"> <li>• <i>NetCount</i> is 2 if both NetA and NetB are up or 1 if only one of the BYNET is up.</li> <li>• <i>Average NetAUse</i> is the sum of all NetAUse of each node divided by the number of online nodes.</li> </ul> This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR PHYSICAL SUMMARY</a> .  <b>Note:</b> NetUse returns a value of zero because resource usage data is not currently available.

Field/ Column Name	Data Type	Description
NetAUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul>
NetBUp	VARCHAR (1) NOT NULL	Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul>
ResLogging	SMALLINT NOT NULL range 0-3600 seconds,	Interval in seconds at which resource usage data is written to one or more active resource usage database tables.
ResMonitor	SMALLINT NOT NULL range 0-3600 seconds	Interval in seconds at which all resource usage data is collected in memory for reporting via the PM/API.
Release	VARCHAR (29) NOT NULL	Release number of the currently running database software (for example, 15.00.00.00). This value is supplied by the database.
Version	VARCHAR (32) NOT NULL	Version number of the currently running database software (for example, 15.00.00.00). This value is supplied by the database.
CollectionDate	DATE NOT NULL	Date the Physical Resource cache was last refreshed.
CollectionTime	FLOAT NOT NULL	Time the Physical Resource cache was last refreshed.

### Sample Input - CLiv2 Request

The following example shows how the parcels for a MONITOR PHYSICAL SUMMARY request, built by CLiv2, look when sent to the database.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.



Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	28	Request	MONITOR PHYSICAL SUMMARY
0003	Data	6	MonVerID	9
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

The following example shows the values returned in character text format for the MONITOR PHYSICAL SUMMARY request. Your application program may display returned values in a different format.

Submitting request MONITOR PHYSICAL SUMMARY; ...

AvgCPU: 100.00 AvgDisk: 10.75 AvgDiskIO: 6536.00

HighCPUUse: 100.00 HighDisk: 100.00 HighDiskIO: 10.75

HighCUProcId: 10001 HighDiskProcId: 10001 HighDiskIOProcId: 10001

LowCPUUse: 10.75 LowDisk: 6536.00 LowDiskIO: 6536.00

LowCUProcId: 10001 LowDiskProcId: 10001 LowDiskIOProcId: 10001

NetUse: 0.00 NetAUp: U NetBUp: U

ResLogging: 60 ResMonitor: 60

Release: 16t.00.00.97 Version: 16t.00.00.97\_dr182707j

Collection Date/Time: 88/64/110954 00:00:00.00

### Warning and Error Messages

All users who are logged on and issue a MONITOR PHYSICAL SUMMARY request after a system restart or after the last rate change can expect to receive a warning. Users will also receive a warning if the resource monitoring rate (ResMonitor) is set to zero.

Either MONITOR PHYSICAL SUMMARY or MONITOR PHYSICAL RESOURCE requests issues a warning for any sessions logged on prior to the database recovery, or prior to the change in the ResMonitor collection rate.

For more detailed information on warning and error messages, see *Teradata Vantage™ - Database Messages*, B035-1096.

### **Relationship Between MONITOR PHYSICAL SUMMARY and MONITOR PHYSICAL CONFIG**

Use the MONITOR PHYSICAL SUMMARY request with the MONITOR PHYSICAL CONFIG request for an overall system status. These are low overhead requests.

- Execute the MONITOR PHYSICAL SUMMARY request every 5 or 10 minutes for a low-cost, continuous monitoring of your system.
- Execute the MONITOR PHYSICAL CONFIG request to get a picture of your system configuration at defined times, such as at the beginning of a day, various times during the day, or when the system is down.

For information on this PMPC CLIv2 or the Teradata JDBC Driver request relationship, see [Relationship Between MONITOR PHYSICAL CONFIG and MONITOR PHYSICAL SUMMARY](#).

### **Relationship Between MONITOR PHYSICAL SUMMARY and SET RESOURCE RATE**

The SET RESOURCE RATE request sets the ResMonitor and ResLogging rates, which are among the responses returned by the MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY request. Any change to either the ResMonitor or ResLogging rate results in changes in the corresponding response returned by the MONITOR VIRTUAL SUMMARY or MONITOR PHYSICAL SUMMARY request.

You must set ResMonitor to a nonzero rate for MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY to return meaningful resource utilization data. A zero ResMonitor rate returns NULL for resource utilization information.

### **Relationship Between MONITOR PHYSICAL SUMMARY and SET SESSION RATE**

Changes to the session-level rates (global and local) specified by SET SESSION RATE are reported in the data returned by MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY.

---

#### **Note:**

The local rate reported is your own local rate. If the local rate is not set, the local rate is reported as zero.

---

As more session-level monitoring is done (by setting a faster SET SESSION RATE), the resulting overhead may increase the level of CPU usage (reported in MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY data) by your system. However, this may depend on the size of the rate change and the type of work done by other sessions.

## **MONITOR SESSION**

Returns session or request resource usage statistics.

## Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 2 or later.  <b>Note:</b> Version 6 or later can be used to determine if a utility session is on the Teradata dynamic workload management software delay queue. For information on returning the utility delay queue, see <a href="#">TDWMGetDelayedUtilities</a> . For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .
<i>host_id</i>	SMALLINT	Logical ID of a host (or client) with sessions logged on. <i>host_id</i> cannot exceed 1023. A <i>host_id</i> of zero identifies the system console ID of the host on which sessions are running.
<i>session_no</i>	INTEGER	Session number. The session number combined with the <i>host_id</i> represents a unique session ID.
<i>user_name</i>	VARCHAR (512)	Name of user who is running the sessions.

- If you do not specify *host\_id*, *user\_name*, or *session\_no*, or *group\_id*, all hosts, all users, or all sessions, or all groups are monitored. For example, if you specify 127 for *host\_id*, PEDERSON for *user\_name*, and do not specify *session\_no* or *group\_id*, the MONITOR SESSION request reports on all sessions currently logged on from host 127 as user PEDERSON.
- NULL in any field, with the exception of *mon\_ver\_id*, indicates a match for all potential values of that field. A NULL for *mon\_ver\_id* will produce an error response. Remember that IndicData parcels are used to specify the output fields that are null. For additional information on IndicData, see [Creating a Request with CLlv2](#).

For information on the data returned for each monitor version, see [Response Groups](#).

## Monitor Privileges

To use this request, you must have MONSESSION privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR SESSION

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

Use this information, when a job hangs because of unavailable resources, to find out about the following:

- The user causing a block
- The locked database or table
- System usage data on a session-by-session basis

In the MONITOR partition, each job is a session. You can execute the MONITOR SESSION request to query:

- All hosts (or clients) with all sessions
- Single host (or client) with all sessions
- Single host (or client) with all sessions for a given user name
- Single host (or client) with a single session and a given user name

MONITOR SESSION requests return data:

- Identical for all sessions
- Specific to a particular session

MONITOR SESSION requests return accumulated session statistics commencing with the last time that one of the following events impacted a session:

- The time that a session logged on.
- The time that a session-switch changed the vproc with Session Control responsibility for a session.
- The time that a system crash or TPA restart occurred.
- The time that system or local rate was set to a nonzero value.

A MONITOR SESSION request can also be used to track session-level AMP usage and provide data on the three highest and lowest CPU/IO utilized AMPs. This can help identify:

- Skewed data
- Possible hardware problems
- Possible database bugs

You can use the data returned by the MONITOR SESSION request as input for the IDENTIFY request to determine lock activity. For more information, see [IDENTIFY](#).

Be aware of the following rules:

- To collect statistics, set the system or local rate to a nonzero value.
- Data on a particular session is reported only for the last *complete* collection period. For example, assume the collection rate is 600 seconds (or 10 minutes) and that you issue a MONITOR SESSION

request at 9:00 a.m. If the user for whom you are seeking data logged onto the system at 9:01 a.m., no session data would be available until the next MONITOR SESSION request is issued at 9:10 a.m. (upon completion of the current collection period). A MONITOR SESSION request made at 9:05 a.m. (after the user logged on at 9:01 a.m.) reports only on sessions that stayed logged on at 9:00 a.m.

- When a session-switch changes the processor with session control responsibility for a session, previous data in memory is lost and the data collection for the affected session starts over again.
- When a system crash occurs, the system crash clears out the previous data and data collection starts over again.

The RSS data loss affects only specific sessions when a session logs off or a PE goes down, forcing all sessions on that PE to switch to another PE. In other cases, such as a system outage, the data loss affects all sessions.

If a system failure occurs, the response to subsequent MONITOR SESSION requests contains a warning message. For more information, see *Teradata Vantage™ - Database Messages*, B035-1096.

The MONITOR SESSION request always returns an ActivityCount (one of the fields in the Success parcel indicating total number of records selected, updated, and so forth) equal to 0 or -1.

If ActivityCount is...	The ...
0	requested combination of logical HostId, UserName, and SessionNo does not match a session.
-1	request generates an unknown number of Response Rows. The application must continue to gather data until it encounters the EndStatement parcel. A value of -1 is used because ActivityCount cannot be determined when the request is executed. Because the response is collected from data shared by all running Monitor sessions, that data can be updated while a request is in progress. If that update adds or removes information on sessions, the activity count calculated at the beginning of a particular session query response generation does not become valid by the end of the response processing for that session. Rather than return possibly incorrect data, the ActivityCount is set to indicate that the count is unknown.

A Trusted Session enables a middle-tier application to switch the user on an already active database session to another user (proxy user). Once the user is switched, all subsequent requests uses the privileges and session attributes of the proxy user. Monitor Session returns the logon user name and ID in the UserName and UserId fields and the proxy user name and ID in the ProxyUserName and ProxyUserId fields.

The following CPU fields in the MONITOR SESSION response are affected by the MonSesCPUNormalization field:

<ul style="list-style-type: none"> <li>• AMPCPUsec</li> <li>• AvgAmpCPUsec</li> <li>• HotAmp1CPU</li> <li>• HotAmp2CPU</li> </ul>	<ul style="list-style-type: none"> <li>• HotAmp3CPU</li> <li>• LowAmp1CPU</li> <li>• LowAmp2CPU</li> <li>• LowAmp3CPU</li> </ul>	<ul style="list-style-type: none"> <li>• PECPUsec</li> <li>• RequestAmpCPU</li> </ul>
---	--	---

The MonSesCPUNormalization field, a DBS Control General Record field, controls whether normalized or non-normalized statistical CPU data is reported by the MONITOR SESSION request, and by the functions: MonitorMySessions and MonitorSession. For more information about the MonSesCPUNormalization field and CPU normalization, see the DBS Control utility in *Teradata Vantage™ - Database Utilities*, B035-1102.

For a complete description of these fields, see [Response Groups](#).

You can also refer to [MonitorMySessions](#) or [MonitorSession](#) for a list of these CPU fields.

## Collecting Session Data

After a collection interval is set to a nonzero rate, session usage data is accumulated in AMP and PE storage areas (regardless of the rate set). When a user makes a MONITOR request, the central coordinator task determines if more current data is necessary. If more current data is required, the central coordinator task directs the processors to transmit data from processor storage areas on the AMPs to the main memory repository on each PE where each PE is simultaneously collecting its own session data. Otherwise, data in the main PE repositories is considered current and is sent to the user.

All current data is associated with an internal timestamp not visible to the user. Every MONITOR SESSION request issued by a user is associated with a session-level data collection rate. This rate is the local rate if a local rate has been set. Otherwise, it is the global rate.

When a user executes a MONITOR SESSION request, the central coordinator task checks the age of the current session-level data (current time minus the data timestamp). Depending on the age determination, action is then taken on the data as shown in the following table.

If data in the PE memory repository is...	The ...
considered current (the age of the data is less than or equal to the session collection rate)	data is returned to the user.
not considered current (the age of the data is greater than the session collection rate)	coordinator task forces a data update (causing its internal timestamp to be reset to the current time) and returns the updated data to the user.

## Unreported Session Partitions

Resource usage in some session partitions may not be reported or fully reported. MONITOR SESSION requests do not report statistics on session resources used by the DBCUTIL utility. Although resource usage is reported, not all resource usage data is accounted for. These restrictions may affect the following data returned from a MONITOR SESSION request:

- AMPCPUSec
- AMPPIO
- Request\_AMPSpool
- AMPState
- PECPUSec
- PEState
- ReqCount

As an example of how the returned data is affected, both AMPState and PESTate data values may be UNKNOWN if I/O is done on behalf of the unreported partitions. As another example, in a Teradata SQL session, PECPUsec spent in a PE do not include time spent in Gateway communications processing.

## A Down Processor

When a previously active PE is down because of a system outage, the data returned for those sessions logged on to the down processor may not be meaningful. If you encounter a system outage, pay attention to the following information, because the PE data does not change:

<ul style="list-style-type: none"> <li>• HostId</li> <li>• LogonPENo</li> <li>• SessionNo</li> <li>• UserName</li> <li>• UserAccount</li> </ul>	<ul style="list-style-type: none"> <li>• UserID</li> <li>• LSN</li> <li>• LogonTime</li> <li>• LogonDate</li> <li>• PartName</li> </ul>	<ul style="list-style-type: none"> <li>• PESTate</li> <li>• LogonSource</li> </ul>
---	---	--

## Internal Sessions

Internal sessions are a part of the database software and cannot be started or aborted from the client. For example, the operation sending a message from a parser to an AMP is an internal session.

Internal sessions can cause problems because:

- They may be blocking an important session.
- They are hard to recognize.
- They may be blocked by other requests.

Internal sessions that are blocked are reported by MONITOR SESSION. These include:

- Internal sessions associated with the user that show the blocking activity on the user session itself.
- DBQL/Teradata dynamic workload management software artificial internal sessions that appear in the Monitor Session output and exist only to show blocked DBQL/Teradata dynamic workload management software internal express requests.

---

### Note:

For information on handling blocked DBQL/Teradata dynamic workload management software internal express requests, see *Teradata Vantage™ - Database Administration*, B035-1093.

---

You must determine what to do about a session that is blocking some important activity.

The following fields (returned by a MONITOR SESSION request) may provide information about internal sessions blocking other sessions:

- Blk\_x\_HostId
- Blk\_x\_SessNo
- Blk\_x\_UserID

Some internal sessions are not easy to recognize simply from the data returned in these Blk\_x fields. With an internal session, any or all of the Blk\_x fields may be NULL. In this case, you might mistake an internal session with a NULL Blk\_x\_SessNo for a client utility session. All three fields could be filled in with legitimate looking values. In this case, you can frequently recognize an internal session by a value of zero in the Blk\_x\_HostId field. Still, this is not guaranteed, because a Teradata SQL session started from the system console running DBW has a value of zero in the Blk\_x\_HostId field.

The following matrix provides some guidelines for recognizing internal sessions the HostId, SessNo, and UserID Blk\_x fields internal sessions.

For fields:	If Values Are:	Session Type is:
HostId, SessNo, UserID	NULL	<b>Internal</b> session
<ul style="list-style-type: none"> <li>HostId &amp; SessNo</li> <li>UserID</li> </ul>	<ul style="list-style-type: none"> <li>NULL</li> <li>Non-NULL</li> </ul>	Idle Client utility
HostId, SessNo, UserID	Non-NULL	Special rule: A MONITOR SESSION request does not return a record for an internal session. If you specify <b>internal</b> session in an IDENTIFY request to specify the name of the session causing the lock, the session returns an error message.

If ...	You ...
an internal session is blocking an important session	cannot abort the internal session.
the internal session lock request is waiting	can abort the work of the sessions that are blocking the internal sessions.
the internal session lock request is granted	must wait for it to complete.

## CLiv2 Response Parcels

The MONITOR SESSION request is treated internally as a multiple statement request with each statement generating a response. The multiple statement response returned from the database contains the following sequence of parcel types:

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments and Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = 1 ActivityType = 84 (PCLMONSESS)
DataInfo	71	6 to 64100	(Optional). This parcel is present if request was IndicData parcel.
Record	10	5 to 64100 (record mode)	Returns data in Record or Indicator mode depending on the request (for example, Data or IndicData).



Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments and Key Parcel Body Fields
		6 to 64100 (indicator mode)	
EndStatement	11	6	StatementNo = 2-byte integer
Success	8	18 to 273	StatementNo = 2 ActivityCount = -1 or 0 ActivityType = 84 (PCLMONSESS)
DataInfo	71	6 to 64100	(Optional) This parcel is present if request was IndicData parcel.
Record	10	5 to 64100 (record mode) 6 to 64100 (indicator mode)	Returns data in Record or Indicator mode depending on the request (for example, Data or IndicData). It contains data describing the session.
EndStatement	11	6	StatementNo = 2-byte integer, value = 2
Success	8	18 to 273	StatementNo = 3 ActivityType = 84 (PCLMONSESS)
DataInfo	71	6 to 64100	(Optional) This parcel is present if request was IndicData parcel.
Record	10	5 to 64100 (record mode) 6 to 64100 (indicator mode)	Returns data in Record or Indicator mode depending on the request (for example, Data or IndicData). It contains blocker data describing the session.
EndStatement	11	6	StatementNo = 2-byte integer, value = 3
EndRequest	12	4	None

## Response

### Note:

Each of the statement types described below correspond to a `ResultSet` returned by the Teradata JDBC Driver, and each statement type field corresponds to a `ResultSet` column. For more information on `ResultSet`s, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Statement 1

The response to the first statement results in a Record parcel containing the following fields:

Field/Column Name	Data Type	Description
CollectionInterval	SMALLINT NOT NULL	Actual interval in seconds between current and last cache refresh.
CollectionSeqNum	INTEGER NOT NULL	Monitor session cache refresh sequence number.
CollectionDate	DATE NOT NULL	Database date of when the session cache was last refreshed.
CollectionTime	FLOAT NOT NULL	Database time of when the session cache was last refreshed.
SessionRate	SMALLINT NOT NULL	Session rate.
ExceptionInterval	SMALLINT NOT NULL	Exception interval. If TASM Workloads are disabled, this field returns a zero. For information on TASM Workloads, see <i>Teradata® Viewpoint User Guide</i> , B035-2206.
SessionRateThreshold	SMALLINT NOT NULL	DBS Control PMPC_SessionRateThreshold value. See the DBS Control PMPC_SessionRateThreshold field in <i>Teradata Vantage™ - Database Utilities</i> , B035-1102 for details.
DBQLFlushRate	SMALLINT NOT NULL	DBS Control DBQLFlushRate value. See the DBS Control DBQLFlushRate field in <i>Teradata Vantage™ - Database Utilities</i> , B035-1102 for details.
RedriveProtection	VARCHAR (2) NOT NULL	Redrive protection type: <ul style="list-style-type: none"> <li>' ' = No Redrive protection. This indicates that the session will not participate in Redrive and database restarts will not be transparent to applications and users.</li> <li>MN = Memory-based Redrive protection, no fallback spools</li> </ul>

## Statement 2

The response to the second statement results in multiple Record parcels that consist of a record for each session in the system. Records are sorted in order of LogonPENo, HostId, and SessionNo.

## Response Groups

There are several groups of response data fields or JDBC ResultSet columns returned by the MONITOR SESSION request. The following table shows the different values returned from *mon\_ver\_id*.

mon_ver_ ID Entry	Response Group Returned	Description
2	Group I Data Fields and JDBC ResultSet Columns	Returns data fields concerned primarily with session-level user status.

mon_ver_ ID Entry	Response Group Returned	Description
3	Groups I Data Fields and II and JDBC ResultSet Columns	In addition, returns data fields on session-specific AMP resource usage.
4	Groups I-III Data Fields and JDBC ResultSet Columns	In addition, returns request level usage information.
5	Groups I-IV Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through IV
6	Groups I-IV Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through IV.  <b>Note:</b> This value is required for Teradata Dynamic Workload Management PM/APIs.
7 and 8	Groups I-V Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through Group V.
9 - 11	Groups I - VI Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through VI.
12	Groups I - VII Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through VII.
13	Groups I - VIII Data Fields and JDBC ResultSet Columns	Returns data fields concerned with Group I through VIII.

**Note:**

If monitor version software ID 10 or later is specified, object fields in the input area can be up to 512 bytes in variable length in host character set format. For more information, see [Impact of Object Name Length on PM/API Requests](#).

**Group I Data Fields and JDBC ResultSet Columns**

The Record returns the following Group I values.

Field/Column Name	Data Type	Description
HostId	SMALLINT	Logical host ID associated with a PE or session. For a PE, the Host ID identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a Host ID and a session number uniquely identifies a user session on the system.  <b>Note:</b> This value is NULL for AMPs. A value of zero represents the Supervisor window.

Field/Column Name	Data Type	Description
LogonPENo	SMALLINT NOT NULL	Vproc number of the PE the session is currently logged on to; it identifies the PE that has control responsibility for the session. Normally, this is the PE that processed the logon request; but if that PE goes offline, it will be the PE to which the session was switched.
RunVprocNo	SMALLINT	Vproc number of the AMP or PE currently assigned to process the session requests. For sessions in Teradata SQL partitions, this value is identical to the LogonPENo. For sessions in FastLoad or MultiLoad partitions, this is the AMP that initially processes the data being loaded. If a RunVprocNo value of -1 in record mode or NULL in indicator mode is returned by MONITOR SESSION for FastLoad, MultiLoad or FastExport sessions, this indicates that the session is in the process of starting up.
SessionNo	INTEGER NOT NULL	Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.
UserName	VARCHAR (128) CHARACTER SET UNICODE NOT NULL	User name of the session.
UserAccount	VARCHAR (128) CHARACTER SET UNICODE NOT NULL	Current account for the session.
UserID	INTEGER NOT NULL	User or internal ID of a user for this session. Within the database, UserID is equivalent to Database ID. Typically, UserID is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, UserID can identify either a given user or database name.
LSN	INTEGER NOT NULL	Logon Sequence Number (LSN) that is associated with a session when the session logs on. It identifies a collection of sessions performing a related activity. For example, in a FastLoad job, a user is logged on as a Teradata SQL session, as well as $n$ FastLoad sessions with the same user name. Therefore, $n+1$ sessions (1 Teradata SQL and $n$ FastLoad) with the same LSN are all associated with the given FastLoad job. To see how the FastLoad job is doing, the user can pick out all sessions reported with the same LSN number.  <b>Note:</b> This information supplies the parent-child relationship for sessions involved with FastLoad and MultiLoad jobs.

Field/Column Name	Data Type	Description
LogonTime	FLOAT NOT NULL	Time portion of the information recorded by Session Control when a session successfully logs on. Together with LogonDate, it indicates when the session logged on to the system. It is usually formatted for display as 99:99:99, which represents hours: minutes: seconds.
LogonDate	DATE NOT NULL	Date portion of the information recorded by Session Control when a session successfully logs on. Together with LogonTime, it indicates when the session logged on to the system.
PartName	VARCHAR (16) NOT NULL	Name of the session partition associated with this session. Following a successful logon request by a session or as part of a connect request, the session identifies the partition with which the user wants the session to be associated. FASTLOAD, Teradata SQL, MONITOR, INTERNAL are examples of valid partition names.
Reserved2	VARCHAR (2)	<b>Note:</b> This field is not currently used.
PEState	VARCHAR (18)	<p>Current state of the session within the PE. This session describes PARSING, DISPATCHING, and Monitor activity. The session states are reported in decreasing priority:</p> <ul style="list-style-type: none"> <li>• <b>DELAYED:</b> The request is either waiting on a queue table for rows to be inserted in to that table or, because of a TASM System Throttle and Session Control rule, the request is on the Teradata dynamic workload management software delay queue. See <i>Teradata® Viewpoint User Guide</i>, B035-2206 for information on this rule.</li> <li>• <b>DEFERRED:</b> A request is being deferred by an arrival rate meter rule.</li> <li>• <b>HOST-RESTART:</b> A restart is in progress for the associated host (or client).</li> <li>• <b>ABORTING:</b> The transaction is being rolled back; the session is aborting.</li> <li>• <b>PARSING-WAIT:</b> Waiting for information from the Data Dictionary.</li> <li>• <b>PARSING:</b> The Parser portion of the PE is processing a request.</li> <li>• <b>ELICIT CLIENT DATA:</b> The Dispatcher is eliciting data from the client and sending it to the AMP.</li> <li>• <b>DISPATCHING:</b> The Dispatcher or Monitor is having a request executed.</li> <li>• <b>BLOCKED:</b> Some background activity is in progress and the last request is on hold until this background activity is completed.</li> <li>• <b>ACTIVE:</b> Normal, on-going activity is being done by this session.</li> <li>• <b>RESPONSE:</b> The Dispatcher is returning query responses to the session.</li> <li>• <b>IDLE: IN-DOUBT:</b> A session using Two-Phase Commit (2PC) is currently <b>IN-DOUBT</b>.</li> <li>• <b>IDLE:</b> No work in progress for this session.</li> <li>• <b>QTDELAYED:</b> A session is delayed due to a Queue Table restriction.</li> </ul>

Field/Column Name	Data Type	Description
		<ul style="list-style-type: none"> <li>• <b>SESDelayed:</b> A utility session is on the Teradata dynamic workload management software delay queue.</li> <li>• <b>UNKNOWN:</b> The Parser, Dispatcher, and Monitor on the PE are unaware of this session.</li> </ul> <p>This value is NULL when a request for data is made before completion of the first collection period that follows either a system outage or a change in the ResMonitor rate.</p>
PECPUsec	FLOAT	<p>CPU time, in seconds, used in a PE by the associated session for parsing and dispatching requests. It is accurate to the second.</p> <p>This value is valid only when associated with Teradata SQL and MONITOR partition sessions.</p> <p>This value is NULL when it is returned for all other sessions.</p>
XActCount	FLOAT	<p>Number of explicit and implicit transactions executed by the session.</p> <p>This value is valid only when returned for Teradata SQL sessions, and is NULL for all other partition sessions. For this value, you must make a request for data before completion of the first collection period that follows either a system outage or a change in the ResMonitor rate.</p>
ReqCount	FLOAT	<p>Number of requests (Tequel Start Request [TSR] messages) initiated by the session.</p> <p>This value is NULL when a request for data is made before completion of the first collection period following either a system outage or change in the ResMonitor rate.</p>
ReqCacheHits	FLOAT	<p>Number of times that this session processed a request using information from the Teradata SQL Parser request cache, specifically, how many times there was a request cache hit.</p> <p>This value is valid only for Teradata SQL sessions, and is NULL for all other partition sessions. This value is also NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.</p>
AMPState	VARCHAR (18)	<p>Current state of the associated session in AMP vprocs in decreasing priority:</p> <ul style="list-style-type: none"> <li>• <b>ABORTING:</b> The transaction is being rolled back; session is aborting.</li> <li>• <b>BLOCKED:</b> Some background activity is in progress and the last request is on hold until this background activity is completed.</li> <li>• <b>ACTIVE:</b> Normal, on-going activity is being done by this session.</li> <li>• <b>IDLE:</b> No work in progress for this session on any AMP.</li> <li>• <b>UNKNOWN:</b> No recorded activity by this session since monitoring began.</li> </ul> <p>This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</p>

Field/Column Name	Data Type	Description
AMPCPUSec	FLOAT	Current elapsed CPU time, in seconds, used on all AMPs by the associated session for executing requests. For example, for Teradata SQL requests, this is the time spent by the database actively working or rolling back an aborted transaction. This does not include any PDE CPU time spent handling database requests. This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.
AMPIO	FLOAT	Current number of logical Reads and Writes issued across all AMPs by the associated session. This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.
Request_AmpSpool	FLOAT	Current spool used by current request across all AMPs, expressed as a number of bytes. This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.
Blk_1_HostId, Blk_2_HostId, Blk_3_HostId	SMALLINT	Logical host ID of a session causing a block. This value is derived from equating the transactions causing a database lock conflict to the sessions that issued those transactions. The Blk_x_HostId in combination with Blk_x_SessNo uniquely identifies the session that is causing a block. This value is NULL if: <ul style="list-style-type: none"> <li>• The host ID is not available.</li> <li>• The session does not have an AMPState of BLOCKED.</li> </ul> If the Blk_x_HostId, Blk_x_SessNo, and Blk_x_UserId values all return as NULLs and AMPState is BLOCKED, a Host Utility (HUT) lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists. Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i> , B035-1102.
Blk_1_SessNo, Blk_2_SessNo, Blk_3_SessNo	INTEGER	Number of the session causing a block. This value is derived from associating the transactions causing a lock conflict to the sessions that issued those transactions. The Blk_x_SessNo in combination with Blk_x_HostId uniquely identifies the session causing a block. This value is NULL if: <ul style="list-style-type: none"> <li>• The SessNo is not available.</li> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> If the Blk_x_HostId, Blk_x_SessNo, and Blk_x_UserId values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The

Field/Column Name	Data Type	Description
		lock was never released, and no blocking information is available because the session no longer exists. Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i> , B035-1102.
Blk_1_UserID, Blk_2_UserID, Blk_3_UserID	INTEGER	ID of the user or host utility job preventing the session from being granted a lock. The user ID is the only information available about who placed the lock. This value is NULL if: <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> If the Blk_x_HostId, Blk_x_SessNo, and Blk_x_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists. Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i> , B035-1102.
Blk_1_LMode, Blk_2_LMode, Blk_3_LMode	VARCHAR (1)	Mode (severity) of the lock involved in causing a block: <ul style="list-style-type: none"> <li>• E = Exclusive</li> <li>• W = Write</li> <li>• R = Read</li> <li>• A = Access</li> </ul> Locks are reported in decreasing order of severity because removing the most severe lock conflict may eliminate the source of the lock conflict. This value is NULL if: <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> A session may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock in the queue and is in conflict with the lock requested by this blocked session. For information on whether the lock is granted, see the block status MONITOR SESSION fields (Blk_1_Status, Blk_2_Status, and Blk_3_Status) later in this table.
Blk_1_OType, Blk_2_OType, Blk_3_OType	VARCHAR(2)	Type of object whose lock is causing the session described by the associated row to be blocked: <ul style="list-style-type: none"> <li>• D = Database</li> <li>• T= Table</li> <li>• R = Row hash</li> <li>• TP = Table Partition range</li> </ul>



Field/Column Name	Data Type	Description
		<ul style="list-style-type: none"> <li>• RP = RowHash in Partition range</li> <li>• RK = RowHash in one partition</li> <li>• RN = RowKey range</li> </ul> <p>However, this object is not necessarily the type of object the blocked session is trying to access. For example, if the session is requesting a row hash lock, the blocking object could be a database, table, or row hash.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> <p>For a Table T, it is possible for User A to block User B with a table level lock on Table T on AMP_1 and with a Row Hash Level lock on that same Table T on AMP_2. When that occurs, the only lock conflict reported is that User B is blocked by User A on a table.</p>
Blk_1_ObjDBId, Blk_2_ObjDBId, Blk_3_ObjDBId	INTEGER	<p>Unique ID of the database object over which a lock conflict is preventing the session from being granted a lock.</p> <p>Within the database system, Database ID is equivalent to User ID. Typically, User ID is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, Database ID can identify either a user or database name.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>
Blk_1_ObjTId, Blk_2_ObjTId, Blk_3_ObjTId	INTEGER	<p>Unique ID of the table object over which a lock conflict is preventing the session from being granted a lock.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• The Blk_x_OType is D.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>
Blk_1_Status, Blk_2_Status, Blk_3_Status	VARCHAR (1)	<p>Status of lock causing a block:</p> <ul style="list-style-type: none"> <li>• W= Waiting</li> <li>• G = Granted</li> </ul> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>

Field/Column Name	Data Type	Description
		<p><b>Note:</b></p> <p>A lock request may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock request in the queue and is in conflict with it.</p> <p>A status of Waiting has a higher priority than that of Granted when there is more than one lock involved. For example, for a given object and a given session, a session that is blocked by a Waiting lock on one AMP and a Granted lock on another AMP has Waiting reported as its status.</p>
MoreBlockers	VARCHAR (1)	<p>Indicator of more lock conflicts:</p> <ul style="list-style-type: none"> <li>Blank = Blk_x information is a complete list of sessions blocking the session described.</li> <li>Asterisk (*) = Additional sessions are blocking the session described. In rare cases, the blocker information in statement 2 and 3 do not represent a complete list of sessions blocking the session described.</li> </ul> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>The state of the session is not BLOCKED.</li> <li>A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>
LogonSource	VARCHAR (128)	<p>Logon source information. At logon time, this information is optionally supplied by the Teradata Director Program or the Gateway to further identify the physical or logical location of the session, the logon user name, and the interface under which the session was initiated. (For example, the data string may include DBC as the user ID and BTEQ as the interface.)</p> <p>Data strings for TCP/IP sessions are inserted by CLv2 or the Teradata JDBC Driver, which truncates strings that exceed 128 bytes.</p> <p><b>Note:</b></p> <p>A two-byte value precedes the LogonSource data to indicate the length of the string. The length value is zero if LogonSource is NULL.</p> <p>For a list of the commonly seen LogonSource string application names, see <i>Teradata Vantage™ - SQL Data Definition Language Detailed Topics</i>, B035-1184.</p>
TempSpace	FLOAT	<p>Total amount, in bytes, of temporary space used by the session. This value is NULL if the session did not materialize any temporary tables.</p>

## Group II Data Fields and JDBC ResultSet Columns

The Record returns the following Group II values.

**Note:**

The values are NULL if the request is made before the collection period expires.

Field/ Column Name	Data Type	Description
HotAmp1CPU	FLOAT	CPU time of the highest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp2CPU	FLOAT	CPU time of the second highest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp3CPU	FLOAT	CPU time of the third highest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.
HotAmp1IO	FLOAT	I/O count of the highest I/O utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp2IO	FLOAT	I/O count of the second highest I/O utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp3IO	FLOAT	I/O count of the third highest I/O utilized AMP for the last collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs in the system.
HotAmp1CPUId	SMALLINT	Vproc ID of the highest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp2CPUId	SMALLINT	Vproc ID of the second highest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp3CPUId	SMALLINT	Vproc ID of the third highest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs in the system.
HotAmp1IOId	SMALLINT	Vproc ID of the highest I/O utilized AMP for the last session collection interval.

Field/ Column Name	Data Type	Description
		This value is NULL if the request is made before the collection period expires.
HotAmp2IOld	SMALLINT	Vproc ID of the second highest I/O utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
HotAmp3IOld	SMALLINT	Vproc ID of the third highest I/O utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs in the system.
LowAmp1CPU	FLOAT	CPU time of the lowest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp2CPU	FLOAT	CPU time of the second lowest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp3CPU	FLOAT	CPU time of the third lowest CPU utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.
LowAmp1IO	FLOAT	I/O count of the lowest I/O utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp2IO	FLOAT	I/O count of the second lowest I/O utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp3IO	FLOAT	I/O count of the third lowest I/O utilized AMP during the collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.
LowAmp1CPUld	SMALLINT	Vproc ID of the lowest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp2CPUld	SMALLINT	Vproc ID of the second lowest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.

Field/ Column Name	Data Type	Description
LowAmp3CPUId	SMALLINT	Vproc ID of the third lowest CPU utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.
LowAmp1IOId	SMALLINT	Vproc ID of the lowest I/O utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp2IOId	SMALLINT	Vproc ID of the second lowest I/O utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires.
LowAmp3IOId	SMALLINT	Vproc ID of the third lowest I/O utilized AMP for the last session collection interval. This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.
UpAMPCount	SMALLINT	Total number of AMPs participating for this session during the last session collection interval. This value is NULL if the request is made before the collection period expires.
AvgAmpCPUsec	FLOAT	Average AMP CPU utilization for the last session collection interval. The average is calculated as the sum of CPU utilization for all AMPs participating divided by the number of online AMPs. This value is NULL if the request is made before the collection period expires.
AvgAmpIOCnt	FLOAT	Average AMP I/O utilization for the last session collection interval. The average is calculated as the sum of I/O utilization for all AMPs participating divided by the number of online AMPs. This value is NULL if the request is made before the collection period expires.

### Group III Data Fields and JDBC ResultSet Columns

The Record returns the following Group III values.

Field/Column Name	Data Type	Description
RequestStartTime	FLOAT	Time of the current request on the session started.
RequestStartDate	DATE	Date of the current request on the session started.
RequestAmpCPU	FLOAT	Total CPU usage by the current SQL request on the session on all AMPs. This value contains proper request-level statistics for DBC

Field/Column Name	Data Type	Description
		<p>/SQL sessions running SQL requests only. Ignore the value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.</p> <p>This field is equivalent to the MonitorSession ReqCPU field.</p>
RequestAmpl/O	FLOAT	<p>Total number of accesses by the current SQL request on the session on all AMPs.</p> <p>This value contains proper request-level statistics for DBC/SQL sessions running SQL requests only. Ignore the value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.</p> <p>This field is equivalent to the MonitorSession ReqIO field.</p>

### Group IV Data Fields and JDBC ResultSet Columns

The Record returns the following Group IV values.

Field/Column Name	Data Type	Description
Request Number	INTEGER	<p>Active request number.</p> <p>If no request is running, this field displays a value of zero or NULL in indicator mode.</p> <p>In some rare cases, in the very early phase of a request in parsing state, when PESTate = PARSING, the request number may not be available and returned as zero or NULL. The active request number will be available on the next collection.</p>
WDId	INTEGER	Workload ID associated with the specified request.
Classification Mode	SMALLINT	<p>Indicator if a running query or session is (or any future queries will be) forced into a WD or not.</p> <p>The classification mode is valid only for DBC/SQL sessions and if the Teradata dynamic workload management software is enabled. If the Teradata dynamic workload management software is enabled, it classifies all incoming queries into a WD.</p> <p>If the Teradata dynamic workload management software is enabled, the value is one of the following:</p> <ul style="list-style-type: none"> <li>• 0 = Automatic</li> <li>• 1 = Manual at Request Level</li> <li>• 2 = Manual at Session Level</li> </ul> <p>If the Teradata dynamic workload management software is disabled and the DBC/SQL session is not a valid session, the value is 255 or NULL.</p>

### Group V Data Field and JDBC ResultSet Columns

The Record returns the following Group V value.

Field/Column Name	Data Type	Description
ProxyUser	VARCHAR (128) CHARACTER SET UNICODE	Name of a proxy user in a trusted session.

## Group VI Data Fields and JDBC ResultSet Columns

The Record returns the following Group VI values.

Field/Column Name	Data Type	Description
CPUDecayLevel	SMALLINT	Current most severe decay level as reached due to CPU usage. <b>Note:</b> Nodes can be at different levels of decay (for example, 0, 1, or 2).
IODecayLevel	SMALLINT	Current most severe decay level as reached due to I/O usage. <b>Note:</b> Nodes can be at different levels of decay (for example, 0, 1, or 2).
TacticalCPUException	INTEGER	Number of nodes that encountered a CPU exception.
TacticalIOException	INTEGER	Number of nodes that encountered an I/O exception.
ReqIOKB	FLOAT	Total logical I/O usage in KB.
ReqPhysIO	FLOAT	Number of physical I/Os.
ReqPhysIOKB	FLOAT	Physical I/O usage in KB.
ReqStepsCompletedCnt	INTEGER	Count of completed steps for the current request. If there is no change in the ReqStepsCompletedCnt field from the previous Monitor Session collection, this indicates that there are no new steps completed.
CurrentRedriveParticipation	VARCHAR (1)	Indicates if the session is participating in Redrive. Sessions that use Redrive can enable or disable the functionality using the REDRIVE reserved query band. Possible values: <ul style="list-style-type: none"> <li>• T = Redrive functionality is enabled (database restarts are transparent to applications and users)</li> <li>• F = Redrive functionality is disabled (database restarts are not transparent to applications and users)</li> </ul>
ReqRedriveSpoolSpace	FLOAT	Persistent spool space for the current request.
BlockerSessionCnt	SMALLINT	Total number of blocker sessions. If there are more than three blocker sessions, this field returns the blocking sessions in one or more record parcels in statement 3.

Field/Column Name	Data Type	Description
		<p><b>Note:</b> This field returns only the first three blocking sessions in statement 2.</p> <p><b>Note:</b> This output parameter is available on monitor software version ID 11 or later.</p>
ReqTblOpBytesIn	FLOAT	<p>The total number of bytes transferred into the database from a foreign server for the current request through one or more table operators.</p> <p><b>Note:</b> The request may involve one or multiple table operator executions. The ReqTblOpBytesIn output parameter shows bytes transferred across all invocations within the request.</p> <p><b>Note:</b> This output parameter is available on monitor software version ID 11 or later.</p>
ReqTblOpBytesOut	FLOAT	<p>The total number of bytes transferred out of the database and into a foreign server for the current request through one or more table operators.</p> <p><b>Note:</b> The request may involve one or multiple table operator executions. The ReqTblOpBytesOut output parameter shows bytes transferred across all invocations within the request.</p> <p><b>Note:</b> This output parameter is available on monitor software version ID 11 or later.</p>

## Group VII Data Fields and JDBC ResultSet Columns

The Record returns the following Group VII values.

Field/Column Name	Data Type	Description
ProxyUserId	INTEGER NULLABLE	The user ID charged for SPOOL and TEMP space if being charged to the proxy user.
Zoneld	INTEGER NULLABLE	The unique identifier of the zone.
ReqHotAmpCPU	FLOAT	<p>The CPU time of the highest CPU utilized AMP during the life of the current request on the session.</p> <p>This value is NULL if there is no request running on the session.</p>



Field/Column Name	Data Type	Description
ReqHotAmpCPUId	SMALLINT	Vproc ID of the highest CPU utilized AMP for the current request. This value is NULL if no request is running on the session.
ReqHotAmpIO	FLOAT	I/O count of the highest I/O utilized AMP during the life of the current request on the session. This value is NULL if there is no request running on the session.
ReqHotAmpIOld	SMALLINT	Vproc ID of the highest I/O utilized AMP for the current request. This value is NULL if there is no request running on the session.
ReqInvolvedAMPCnt	SMALLINT	The number of AMPs involved in processing the current request. This value is NULL if there is no request running on the session.
ReqFirstRespTime	FLOAT	Time that the first response of the current request on the session is ready. The response may be held to meet the TASM Minimum Response Time. This value is NULL if there is no request running on the session or the PE state is not RESPONSE.
ReqFirstRespDate	DATE	Date that the first response of the current request on the session is ready. The response may be held to meet the TASM Minimum Response Time. This value is NULL if there is no request running on the session or the PE state is not RESPONSE.
ReqLocalQueryStatus	SMALLINT	The current state of the Unified Data Architecture (UDA) query. This value is NULL when no UDA query is running.
ReqRemoteHostId	SMALLINT	Host ID of the remote system. This value is NULL when there is no UDA query running.
ReqRemoteSessionId	INTEGER	Session ID of the executing remote query. This value is NULL when there is no UDA query running.
ReqRemoteRequestId	INTEGER	Request ID of the executing remote query. This value is NULL when there is no UDA query running.
ReqRemoteQueryId	FLOAT	Query ID of the executing remote query. This value is NULL when there is no UDA query running.

### Group VIII Data Fields and JDBC ResultSet Columns

The Record returns the following Group VIII values.

Field/Column Name	Data Type	Description
ReqHotAmpSpool	FLOAT	Spool value of the highest spool utilized AMP during the life of the current request on the session. This value is NULL if no request is running on the session.

Field/Column Name	Data Type	Description
ReqHotAmpSpoolId	SMALLINT	Vproc ID of the highest spool utilized AMP for the current request. This value is NULL if no request is running on the session.
ReqMapNo	SMALLINT	Map number for the largest map the request is using.
ReqMaxNumMapAMPs	INTEGER	Number of AMPs in the largest contiguous map used by the request.
ReqMinNumMapAMPs	INTEGER	Number of AMPs in the smallest contiguous map used by the request.
ReqSysDefNumMapAMPs	INTEGER	Number of AMPs in the system-default map used by the request.
ReqRemoteHostIp	VARCHAR(128)	Host IP address of the remote system.
ReqServerName	VARCHAR(128)	Name of the foreign server.
ReqFlexReleased	SMALLINT	The TDWM Flex Throttle feature detects available system resources, overrides existing workload throttle limits and automatically releases queries from the delay queue. This minimizes the DBA manually managing the TASM delay queue. Note, only Workload throttles are overridden; all System level throttles are still honored. Return values: <ul style="list-style-type: none"> <li>• 0: Indicates that the request was not released by TDWM Flex Throttles.</li> <li>• 1: Indicates that the request was released by TDWM Flex Throttles.</li> </ul>
ReqAdmissionTime	FLOAT	Time that the current request was admitted into the system by workload management (after being checked for applicable arrival rate meters).
ReqAdmissionDate	DATE	Date that the current request was admitted into the system by workload management (after being checked for applicable arrival rate meters).

### Statement 3

This statement returns additional blocker sessions. If more than three sessions are blocking the session, the first three blocker sessions are listed in statement 2 and the remaining blocker sessions are listed in statement 3. Each record parcel holds one blocker information set. Record parcels are returned in statement 3 for a specified session that has more than three blocking sessions.

For more information about the first three blocker sessions, see [Statement 2](#).

Field/ Column Name	Data Type	Description
HostId	SMALLINT	Logical host ID of the blocked session.
SessionNo	INTEGER, NOT NULL	Session number of the blocked session that has more than three blocker sessions. The first three blocker sessions are listed in statement 2 and the remaining blocker sessions are listed in statement 3.
Blk_HostId	SMALLINT	<p>Logical host ID of a session causing a block. This value is derived from equating the transactions causing a database lock conflict to the sessions that issued those transactions. The Blk_HostId in combination with Blk_SessNo uniquely identifies the session that is causing a block.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The host ID is not available.</li> <li>• The session does not have an AMPState of BLOCKED.</li> </ul> <p>If the Blk_HostId, Blk_SessNo, and Blk_UserID values all return as NULLs and AMPState is BLOCKED, a Host Utility (HUT) lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.</p> <p>Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</p>
Blk_SessNo	INTEGER	<p>Number of the session causing a block. This value is derived from associating the transactions causing a lock conflict to the sessions that issued those transactions. The Blk_SessNo in combination with Blk_HostId uniquely identifies the session causing a block.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The SessNo is not available.</li> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> <p>If the Blk_HostId, Blk_SessNo, and Blk_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.</p> <p>Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</p>
Blk_UserID	INTEGER	<p>ID of the user or host utility job preventing the session from being granted a lock. The user ID is the only information available about who placed the lock.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>

Field/ Column Name	Data Type	Description
		<p>If the Blk_HostId, Blk_SessNo, and Blk_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.</p> <p>Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</p>
Blk_LMode	VARCHAR (1)	<p>Mode (severity) of the lock involved in causing a block:</p> <ul style="list-style-type: none"> <li>• E = Exclusive</li> <li>• W = Write</li> <li>• R = Read</li> <li>• A = Access</li> </ul> <p>Locks are reported in decreasing order of severity because removing the most severe lock conflict may eliminate the source of the lock conflict.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> <p>A session may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock in the queue and is in conflict with the lock requested by this blocked session. For information on whether the lock is granted, see the Blk_Status field.</p>
Blk_OType	VARCHAR(2)	<p>Type of object whose lock is causing the session described by the associated row to be blocked:</p> <ul style="list-style-type: none"> <li>• D = Database</li> <li>• T= Table</li> <li>• R = Row hash</li> <li>• TP = Table Partition range</li> <li>• RP = RowHash in Partition range</li> <li>• RK = RowHash in one partition</li> <li>• RN = RowKey range</li> </ul> <p>However, this object is not necessarily the type of object the blocked session is trying to access. For example, if the session is requesting a row hash lock, the blocking object could be a database, table, or row hash.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> <p>For a Table T, it is possible for User A to block User B with a table level lock on Table T on AMP_1 and with a Row Hash Level lock on that same Table T on Blk_LMode. When that occurs, the only lock conflict reported is that User B is blocked by User A on a table.</p>

Field/ Column Name	Data Type	Description
Blk_ ObjDBId	INTEGER	<p>Unique ID of the database object over which a lock conflict is preventing the session from being granted a lock.</p> <p>Within the database system, Database ID is equivalent to User ID. Typically, User ID is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, Database ID can identify either a user or database name.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>
Blk_ObjTId	INTEGER	<p>Unique ID of the table object over which a lock conflict is preventing the session from being granted a lock.</p> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• The Blk_OType is D.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul>
Blk_Status	VARCHAR (1)	<p>Status of lock causing a block:</p> <ul style="list-style-type: none"> <li>• W= Waiting</li> <li>• G = Granted</li> </ul> <p>This value is NULL if:</p> <ul style="list-style-type: none"> <li>• The session does not have an AMPState of BLOCKED.</li> <li>• A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.</li> </ul> <p><b>Note:</b></p> <p>A lock request may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock request in the queue and is in conflict with it.</p> <p>A status of Waiting has a higher priority than that of Granted when there is more than one lock involved. For example, for a given object and a given session, a session that is blocked by a Waiting lock on one AMP and a Granted lock on another AMP has Waiting reported as its status.</p>

### Sample Input - CLlv2 Request

This example shows how the parcels for a MONITOR SESSION request, built by CLlv2, appear when sent to the database server using a *host\_id* of 387, a *session\_no* of 1098, and a *user\_name* of WEEKLY. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	19	Request	MONITOR SESSION
0003	Data	42	MonVerID	10
			HostId	387
			SessionNo	1098
			UserName	WEEKLY
0004	Resp	6	BufferSize	64000

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output - MONITOR SESSION Statements 1 through 3

Using monitor software version ID 13, this request returns the following values for statements 1 through 3.

```

HostId:      1    SessionNo: 1006
LogonPENO: 30718  RunVprocNo: 30718
PartName: DBC/SQL                                PESTate:  DISPATCHING

LogonDate/Time: 04/29/2016 11:26:46.00
UserID: 1028 LSN: 0
UserName: [JCK]
UserAccount: [DBC]

PECPUSec:      0.02    XactCount:      1.00
ReqCount:      2.0    ReqCacheHits:      0.0

AMPState: ACTIVE
AMPCPUSec:      13.60    AMPIO: 4606.00
Request_AMPSPool: 491517952.0

Blk_1_HostId:      0    Blk_2_HostId:      0    Blk_3_HostId:      0
Blk_1_SessNo:      0    Blk_2_SessNo:      0    Blk_3_SessNo:      0
Blk_1_UserID:      0    Blk_2_UserID:      0    Blk_3_UserID:      0
Blk_1_LMode:      Blk_2_LMode:      Blk_3_LMode:
Blk_1_OLType:      Blk_2_OLType:      Blk_3_OLType:
Blk_1_ObjDBId:      0    Blk_2_ObjDBId:      0    Blk_3_ObjDBId:      0

```

```

Blk_1_ObjTId:      0   Blk_2_ObjTId:      0   Blk_3_ObjTId:      0
Blk_1_Status:      Blk_2_Status:      Blk_3_Status:
MoreBlockers:

LogonSource: (TCP/IP) f1a3 153.64.183.39 MYSYSTEM      9964 JK121219 BTEQ
01 LSS

HotAmp1CPU:        3.43   HotAmp2CPU:        3.40   HotAmp3CPU:        3.40
HotAmp1CPUId:      3     HotAmp2CPUId:      2     HotAmp3CPUId:      0

HotAmp1IO:         1158.00   HotAmp2IO:         1152.00   HotAmp3IO:         1152.00
HotAmp1IOId:       3     HotAmp2IOId:       2     HotAmp3IOId:       1

LowAmp1CPU:        3.38   LowAmp2CPU:        3.40   LowAmp3CPU:        3.40
LowAmp1CPUId:      1     LowAmp2CPUId:      2     LowAmp3CPUId:      0

LowAmp1IO:         1144.00   LowAmp2IO:         1152.00   LowAmp3IO:         1152.00
LowAmp1IOId:       0     LowAmp2IOId:       1     LowAmp3IOId:       2

AvgAmpCPUsec:      3.40   AvgAmpIOCnt:      1151.50
AmpCount:          4

TempSpaceUsg:      0.00

ReqStartTime:      04/29/2016 11:26:46.00
ReqCPU:            13.60   ReqIO:            4606.00
ReqNo:3   WlcId: 13     DontReclassifyFlag: 0
ProxyUser: []

ProxyUserID: 0
CPUDecayLevel=0,   IODecayLevel=0,   TacticalCPUException=0, TacticalIOException=0
ReqIOKB=541295     , ReqPhysIO=661     , ReqPhysIOKB=268502

ReqStepsCompletedCnt=9
CurrentRedriveParticipation=[F]
ReqRedriveSpoolSpace=0.00

BlockerSessionCnt = 0
ReqTblOpBytesIn = 0     , ReqTblOpBytesOut = 0

ZoneId = 0
ReqHotAmpCPU =        3.43   ReqHotAmpCPUId = 3
ReqHotAmpIO =        1158.00   ReqHotAmpIOId = 3
ReqInvolvedAMPCnt = 4

```

```

ReqFirstResp Date/Time= 00/00/0000 00:00:00.00

ReqLocalQueryStatus = 0
ReqRemoteHostId = 0, ReqRemoteSessionId = 0
ReqRemoteRequestId = 0, ReqRemoteQueryId = 0
ReqRemoteHostIp:

ReqServerName:

ReqHotAmpSpool = 123837952.00 ReqHotAmpSpoolId = 3
ReqMapNo = 0, ReqMaxNumMapAMPs = 0
ReqMinNumMapAMPs = 0, ReqSysDefNumMapAMPs = 0
ReqFlexReleased = N (0)

```

### Relationship Between MONITOR SESSION and ABORT SESSION

When sessions are being aborted or sessions are being blocked by the sessions being aborted, data returned from subsequent MONITOR SESSION queries may be affected. After the abort operation starts, you can immediately notice the changes from aborting sessions. However, you will not notice the changes resulting from sessions that were blocked by aborting sessions in MONITOR SESSION responses until the abort operation is complete.

For information on this PMPC PM/API request relationship, see [Relationship Between ABORT SESSION and MONITOR SESSION](#).

### Relationship Between MONITOR SESSION and IDENTIFY

PM/API can report on locks placed by any user or object with the MONITOR SESSION and IDENTIFY requests. The MONITOR SESSION request helps you identify the types of locks blocking a session.

For information on this PMPC PM/API request relationship, see [Relationship Between IDENTIFY and MONITOR SESSION](#).

### Relationship Between MONITOR SESSION and SET SESSION RATE

You must execute the SET SESSION RATE request to activate session-level data collection before you execute a MONITOR SESSION request. This means that either the global rate or local rate must be set to nonzero. If both rates are set to zero, an error message is returned.

A change in the global session rate may affect the data reported by a MONITOR SESSION request. Specifically, if User A makes a change in the global rate, this change could affect the viewing for User B of displayed data from a MONITOR SESSION request. Unless User B is aware of the rate change, User B may draw incorrect conclusions from the observed data. In this case, User B receives a warning message when executing a MONITOR SESSION request.



### Example: Using SET SESSION RATE and MONITOR SESSION

The following example is a single user on a system that uses the SET SESSION RATE and MONITOR SESSION requests.

1. User Morris uses the SET SESSION RATE request to set a session-level global rate of 600 seconds (or 10 minutes) at 9:00 a.m. The data time-stamp was set at 9:00 a.m. because that was the last time data was requested.
2. Morris then uses the SET SESSION RATE request to set a session-level local rate of 300 seconds (5 minutes) at 9:05 a.m. Morris does not make a request for data at this time. Therefore, the time-stamp is still set at 9:00 a.m.
3. Morris executes a MONITOR SESSION request at 9:08 a.m. for *host\_id* of 510 and a *session\_no* of 1000:  
  
MONITOR SESSION 2 510 1000
4. A collection occurs because the “current” data was not considered current (that is, the age of the data is greater than the session collection rate). The system calculates that 9:08 a.m. (current time) - 9:00 a.m. (time-stamp) = 8 minutes, which is the age of the data. The 8 minutes is greater than 5 minutes, which is the local collection rate. Because of the forced update, Morris receives 8 minutes of data (from 9:00 a.m. to 9:08 a.m.), and the time-stamp is reset to 9:08 a.m.
5. At 9:10 a.m., Morris decides to make another MONITOR SESSION request. This time, no collection occurs, because the “current” available data was considered current. The system calculates that 9:10 a.m. (current time) - 9:08 a.m. (time-stamp) = 2 minutes (age of the data), which is less than 5 minutes (local collection rate). This means that the current data available is considered current, because no data update to the session-level memory repository occurs. Morris gets the same data that was returned at 9:08 a.m.

Two other events can cause data to be collected:

Event	Comments
A default system timer of 10 minutes has elapsed without a request from a user.	Whenever 10 minutes has expired without a collection, the system timer causes a default collection to occur. This causes data in the main memory repository to be updated, even if no request is made.
When the AMP Session Cache is full	A full AMP Session Cache forces an update of data in the main memory repository. This is a rare occurrence.

## MONITOR SQL

Returns the step information (that is, a scaled-down version of the output of the EXPLAIN request modifier) of the current or running request for the specified host, session, and vproc.

## Input Data

Element	Data Type	Description
<i>IndByte</i>	BYTE	Indicator bits that specify which fields to treat as NULL if you are using indicator mode. Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.  <b>Note:</b> The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.
<i>mon_ver_id</i>	SMALLINT NOT NULL	MONITOR software version ID. This can be version 3 or later. For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .
<i>host_id</i>	SMALLINT NOT NULL	Logical ID of a host (or client) with sessions logged on. <i>host_id</i> cannot exceed 1023 bytes. A <i>host_id</i> of zero identifies the system console ID of the host on which the sessions are running.
<i>session_no</i>	INTEGER NOT NULL	Session number. The session number combined with the <i>host_id</i> represents a unique session ID.
<i>RunPEVprocNo</i>	SMALLINT NOT NULL	PE vproc number where the session runs. This is typically obtained from the MONITOR SESSION response of the RunVprocNo field. See the MONITOR SESSION RunVprocNo field for more information.

## Monitor Privileges

To use this request, you must have the MONSESSION privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR SQL

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

When issuing a MONITOR SQL request, you must specify a minimum buffer size of 32,007 bytes or more. For some custom applications, an error message may appear.

The MONITOR SQL request can be used in the following ways:

- A problematic query is utilizing a large amount of system resources. Use of this feature provides the associated SQL text to the system administrator to help identify possible bottlenecks, hardware problems and bugs. The query text and DDL statements can also be forwarded to Teradata Support Center for help in diagnosing the problem.
- The information provided by this request can help to identify specific SQL problems, such as poorly structured tables and indexes.
- Problems can even be traced down to the individual step within a query.

The MONITOR SQL request may not be used on internal sessions or sessions that are logged onto the monitor. Request text and steps text are not stored for these types of sessions, and if an attempt is made to query one of them, an error is returned indicating that it is not an SQL session.

Abort processing is handled in the same way as when using the MONITOR SESSION request. For further information about abort handling, see [MONITOR SESSION](#).

If MONITOR SQL processing is not completed within the timeout interval, an error is returned to the client application. When a MONITOR SQL request is timed out, the processing continues internally to its completion. If the client application submits a new MONITOR SQL request for the same timed out target session while the previous timed out one is still being processed, an error is returned.

The timeout interval can be set in the DBS Control field, PMPC\_TimeoutSecs. The default timeout interval is 60 seconds. If the PMPC\_TimeoutSecs field is set to zero, the MONITOR SQL timeout request will be disabled and no timeout will occur. For more information on the PMPC\_TimeoutSecs field, see Utilities.

MONITOR SQL is most useful when used with the MONITOR VIRTUAL SUMMARY request for doing a quick overall system health check. For more information, see [Relationship Between MONITOR VIRTUAL CONFIG and MONITOR VIRTUAL SUMMARY](#).

## CLlv2 Response Parcels

The MONITOR SQL request is treated internally as a three statement request with each statement generating a response. The three-statement response returned from the database contains the following sequence of parcel types:

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 1 ActivityCount = -1 ActivityType = 110 (PCLMONSQL)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the text of the SQL request (see <a href="#">Statement 1</a> ).
EndStatement	11	6	StatementNo = 2-byte integer

Parcel Sequence	Parcel Flavor	Length (Bytes)	Comments/Key Parcel Body Fields
Success	8	18 to 273	StatementNo = 2 ActivityCount = 1 ActivityType = 110 (PCLMONSQL)
DataInfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. This record contains step count information (see <a href="#">Statement 2</a> ).
EndStatement	11	6	StatementNo = 2-byte integer
Success	8	18 to 273	StatementNo = 3 ActivityCount = Number of EXPLAIN steps ActivityType = 110 (PCLMONSQL)
Datainfo	71	6 to 64100	Optional; this parcel is present if request was IndicData parcel.
Record	10	<ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul>	Depending on request (Data or IndicData), data is in record or indicator mode. Each record contains step resource information (see <a href="#">Statement 3</a> ).
EndStatement	11	6	StatementNo = 2-byte integer
EndRequest	12	4	None

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Statement 1

The response to the first statement results in a Record parcel that contains the SQL request text.

Field/Column Name	Data Type	Description
RequestText	VARCHAR (64000)	Actual SQL request for a specified session.

Field/ Column Name	Data Type	Description
		<p><b>Note:</b></p> <p>If <i>RequestText</i> is 64001 bytes or greater in length, it will automatically split into multiple records. A variable number of characters up to 64000 is returned depending on the character set of the session. If needed, the request context is split into multiple rows such that each row does not exceed 64000 bytes.</p>

## Statement 2

The response to the second statement returns information regarding the total number of steps and which steps are currently executing.

Field/ Column Name	Data Type	Description
NumOfSteps	SMALLINT NOT NULL	<p>Number of steps contained in the description text in the third statement of the response.</p> <p>If this is a static plan (that is, when the <i>DynamicPlan</i> field value is zero), <i>NumOfSteps</i> is the total number of steps for the static plan.</p> <p>If this is a complete dynamic plan (that is, when the <i>DynamicPlan</i> field value is 1 and the <i>PartialSteps</i> field value is zero), <i>NumOfSteps</i> is the total number of steps for the dynamic plan.</p> <p>If this is a partial dynamic plan (that is, when both of the <i>DynamicPlan</i> and <i>PartialSteps</i> field values are 1), <i>NumOfSteps</i> is the total number of steps generated. This value is less than the total number of steps generated for the entire dynamic plan.</p> <p>If this is a request with a dynamic plan that has been throttled and is in the delay queue (that is, when the <i>DynamicPlan</i> and <i>PartialSteps</i> field values are 1 and no rows are returned in response to the third statement), <i>NumOfSteps</i> is zero.</p> <p>For more information, see the MONITOR SQL <i>DynamicPlan</i> and <i>PartialSteps</i> fields.</p>
CurLev1StepNum	SMALLINT NOT NULL	<p>Number of the currently executing level 1 step. If parallel steps are executing, it is the number of the lowest executing step.</p> <p>If this is a request with a dynamic plan that has been throttled and is in the delay queue (for example, when the <i>NumOfSteps</i> field value is zero and both the <i>DynamicPlan</i> and <i>PartialSteps</i> field values are 1), the <i>CurLev1StepNum</i> field value is zero.</p>
CurLev2StepNum	SMALLINT NOT NULL	<p>Number of the currently executing step. If parallel steps are executing, it is the number of the highest executing step.</p> <p>If this is a request with a dynamic plan that has been throttled and is in the delay queue (for example, when the <i>NumOfSteps</i> field value is zero and both the <i>DynamicPlan</i> and <i>PartialSteps</i> field values are 1), the <i>CurLev2StepNum</i> field value is 1.</p>

Field/ Column Name	Data Type	Description
DynamicPlan	SMALLINT	Plan type: <ul style="list-style-type: none"> <li>• 0 = Static plan</li> <li>• 1 = Dynamic plan</li> </ul> For more information on static and dynamic explanations of a request, see the EXPLAIN request modifier in <i>Teradata Vantage™ - SQL Data Manipulation Language</i> , B035-1146 or <i>Teradata Vantage™ - SQL Request and Transaction Processing</i> , B035-1142.
PartialSteps	SMALLINT	Possible values: <ul style="list-style-type: none"> <li>• 0 = All steps are returned</li> <li>• 1 = Partial plan or no plan is returned</li> </ul> If a partial plan is returned, this indicates the steps for the final plan fragment of the dynamic explanation of the request has not yet been generated. If no plan is returned, this indicates the request has been throttled and is in the delay queue.  <b>Note:</b> A value 1 cannot occur for a static plan. For more information on static and dynamic explanations of a request, see <i>Teradata Vantage™ - SQL Data Manipulation Language</i> , B035-1146 or <i>Teradata Vantage™ - SQL Request and Transaction Processing</i> , B035-1142.
Zoneld	INTEGER NULLABLE	The unique identifier of the zone.
SPName	VARCHAR(128) CHARACTER SET UNICODE	This is the outer stored procedure name, if a stored procedure is being executed. NULL is returned in indicator mode if no stored procedure is being executed.
SPDBName	VARCHAR(128) CHARACTER SET UNICODE	This is the owner database name of the outer stored procedure, if a stored procedure is being executed. NULL is returned in indicator mode if no stored procedure is being executed.
DefaultDBName	VARCHAR(128) CHARACTER SET UNICODE NOT NULL	This field returns the default database name of the session at the start of a non-stored procedure request. For stored procedures, it returns the default database name of the session when the stored procedure was compiled.

**Note:**

If only one step is executing, CurLevlStepNum and CurLev2StepNum are identical.

### Statement 3

The response to the third statement returns a Record parcel that contains the step text for a given request. Each step returns a separate row. This parcel description has changed from the version 3 parcel. The response parcel received is contingent on whether you set *mon\_ver\_id* 3 or *mon\_ver\_id* 4. The values returned vary in format and content depending on the value of *mon\_ver\_id* used.

If *mon\_ver\_id* 3 is set, the response to the third statement returns a Record parcel that contains the following steps text for a given request.

Field/Column Name	Data Type	Description
StepNum	SMALLINT NOT NULL	Unique number identifying the EXPLAIN step.
StepText	VARCHAR (2048) CHARACTER SET UNICODE NOT NULL	Generated text of the step.

If *mon\_ver\_id* 4 is set, the response to the third statement returns a Record parcel that contains the following steps text for a given request.

Field/Column Name	Data Type	Description
StepNum	SMALLINT NOT NULL	Unique number identifying the EXPLAIN step.
Confidence	SMALLINT NOT NULL	Confidence level as determined by the optimizer: <ul style="list-style-type: none"> <li>• 0 = None</li> <li>• 1= Foreign Key</li> <li>• 2 = Low</li> <li>• 3 = High</li> </ul>
EstRowCount	FLOAT NOT NULL	Estimated row count generated from the Optimizer plan for this step.  For a hybrid join method, also called a partial redistribution and partial duplication (PRPD) plan, the EstRowCount field for the split step (that is, a RETRIEVE or JOIN step with “split into” appearing in the EXPLAIN when target spools are generated) is the estimated row counts for all split spools.  For more information on PRPD, see <i>Teradata Vantage™ - SQL Request and Transaction Processing</i> , B035-1142 or <i>Teradata Vantage™ - Database Administration</i> , B035-1093.
ActRowCount	FLOAT NOT NULL	Actual row count returned from the AMP for this step. For a PRPD plan, it includes rows from all split spools for a split step.  For more information on PRPD, see <i>Teradata Vantage™ - SQL Request and Transaction Processing</i> , B035-1142 or <i>Teradata Vantage™ - Database Administration</i> , B035-1093.

Field/Column Name	Data Type	Description
EstElapTime	FLOAT NOT NULL	Estimated time for the query as generated from the Optimizer plan.
ActElapTime	FLOAT NOT NULL	Actual elapsed time calculated by the dispatcher.
StepText	VARCHAR (2048) CHARACTER SET UNICODE NOT NULL	Generated text of the step.

The estimated fields populate immediately. The low value supplied by the optimizer is used. The actual fields are populated at the same time with a "-1". After the step executes, the actual values (or a zero for those kinds of steps with no row count) are placed in the actual fields to identify those steps that are executed at the time of the API information capture.

**Note:**

If the request has been throttled and is in the delay queue (that is, when the NumOfSteps field value is zero and both the DynamicPlan and PartialSteps field values are 1), the response to this statement returns no rows. For more information on the NumOfSteps, DynamicPlan, and the PartialSteps fields, see [Statement 2](#).

### Sample Input - CLIV2 Request

This example shows how the parcels for a MONITOR SQL request, built by CLIV2, appear when sent to the database server. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

Flavor		Length	Body	
Num	Name	Bytes	Field	Value
0001	Req	16	Request	MONITOR SQL
0003	Data	12	MonVerID HostId SessionNo RunPEVprocNo	3 1 1002 16383
0004	Resp	6	BufferSize	64000



## Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Sample Output

This example shows the values returned in character text format for the MONITOR SQL request. Your application program may display returned values in a different format.

### Note:

The EXPLAIN steps shown are not as inclusive as those provided by the EXPLAIN request modifier.

The following is an Explain of a CREATE DATABASE SQL statement.

```
EXPLAIN CREATE DATABASE testdb AS PERM=1E6;
```

Explanation

- ```
-----
```
- 1) First, we lock data base testdb for exclusive use.
  - 2) Next, we lock DBC.UIFDEPENDENCY for write on a reserved RowHash to prevent global deadlock.
  - 3) We lock DBC.DataBaseSpace for write on a reserved RowHash to prevent global deadlock.
  - 4) We lock DBC.Parents for write on a reserved RowHash to prevent global deadlock.
  - 5) We lock DBC.Owners for write on a reserved RowHash to prevent global deadlock.
  - 6) We lock DBC.AccessRights for write on a reserved RowHash in all partitions to prevent global deadlock.
  - 7) We lock DBC.UIFDEPENDENCY for write, we lock DBC.DataBaseSpace for write, we lock DBC.Parents for write, we lock DBC.Owners for write, we lock DBC.DBase for write on a RowHash, we lock DBC.DBase for write on a RowHash, we lock DBC.Accounts for write on a RowHash, and we lock DBC.AccessRights for write.
  - 8) We execute the following steps in parallel.
    - 1) We do a single-AMP ABORT test from DBC.DBase by way of the unique primary index "Field\_1 = 'TESTDB'" with no residual conditions.
    - 2) We do a single-AMP ABORT test from DBC.Roles by way of the unique primary index "Field\_1 = 'TESTDB'" with no residual conditions.
    - 3) We do a single-AMP ABORT test from DBC.DBase by way of the unique primary index "Field\_1 = 'DBC'" with a residual

```

condition of ("DBC.DBase.Field_18 = 47").
4) We do a single-AMP ABORT test from DBC.DBase by way of the
   unique primary index "Field_1 = 'DBC'" with a residual
   condition of ("1.00000000000000E 006 <= DBC.DBase.Field_10").
5) We do an INSERT into DBC.DBase.
6) We do a single-AMP UPDATE from DBC.DBase by way of the unique
   primary index "Field_1 = 'DBC'" with no residual conditions.
7) We do a single-AMP RETRIEVE step from DBC.Parents by way of
   the primary index "Field_1 = '00000100'XB" with no residual
   conditions into Spool 1 (all_amps), which is redistributed by
   the hash code of (DBC.Parents.Field_2) to few AMPs. Then we
   do a SORT to order Spool 1 by row hash.
9) We do an all-AMPs MERGE into DBC.Owners from Spool 1 (Last Use).
10) We execute the following steps in parallel.
    1) We do an INSERT into DBC.Owners.
    2) We do a single-AMP RETRIEVE step from DBC.Parents by way of
       the primary index "Field_1 = '00000100'XB" with no residual
       conditions into Spool 2 (all_amps), which is redistributed by
       the hash code of ('00000404'XB) to few AMPs. Then we do a
       SORT to order Spool 2 by row hash.
11) We do an all-AMPs MERGE into DBC.Parents from Spool 2 (Last Use).
12) We execute the following steps in parallel.
    1) We do an INSERT into DBC.Parents.
    2) We do an INSERT into DBC.Accounts.
    3) We do a single-AMP RETRIEVE step from a single partition of
       DBC.AccessRights by way of the primary index "Field_1 =
       '00000000'XB, Field_2 = '00000000'XB, Field_3 =
       '000000000000'XB" with a residual condition of (
       "(DBC.AccessRights.Field_5 <> 'ZO') AND
       ((DBC.AccessRights.Field_5 <> 'DZ') AND
       ((DBC.AccessRights.Field_5 <> 'CZ') AND
       ((DBC.AccessRights.Field_5 <> 'OR') AND
       ((DBC.AccessRights.Field_5 <> 'OA') AND
       ((DBC.AccessRights.Field_5 <> 'DS') AND
       ((DBC.AccessRights.Field_5 <> 'CS') AND
       ((DBC.AccessRights.Field_5 <> 'OD') AND
       ((DBC.AccessRights.Field_5 <> 'OU') AND
       ((DBC.AccessRights.Field_5 <> 'OS') AND
       ((DBC.AccessRights.Field_5 <> 'OI') AND
       ((DBC.AccessRights.Field_5 <> 'SA') AND
       ((DBC.AccessRights.Field_5 <> 'SD') AND
       ((DBC.AccessRights.Field_5 <> 'GM') AND
       ((DBC.AccessRights.Field_5 <> 'GD') AND
       ((DBC.AccessRights.Field_5 <> 'GC') AND

```

```

((DBC.AccessRights.Field_5 <> 'OP') AND
((DBC.AccessRights.Field_5 <> 'AE') AND
((DBC.AccessRights.Field_5 <> 'CE') AND
((DBC.AccessRights.Field_5 <> 'DO') AND
((DBC.AccessRights.Field_5 <> 'CO') AND
((DBC.AccessRights.Field_5 <> 'DR') AND
((DBC.AccessRights.Field_5 <> 'CR') AND
((DBC.AccessRights.Field_5 <> 'AP') AND
((DBC.AccessRights.Field_5 <> 'UM') AND
((DBC.AccessRights.Field_5 <> 'UT') AND
((DBC.AccessRights.Field_5 <> 'UU') AND
((DBC.AccessRights.Field_5 <> 'SH') AND
((DBC.AccessRights.Field_5 <> 'EF') AND
((DBC.AccessRights.Field_5 <> 'AF') AND
((DBC.AccessRights.Field_5 <> 'CF') AND
((DBC.AccessRights.Field_5 <> 'PE') AND
((DBC.AccessRights.Field_5 <> 'NT') AND
((DBC.AccessRights.Field_5 <> 'PC') AND
((DBC.AccessRights.Field_5 <> 'TH') AND
((DBC.AccessRights.Field_5 <> 'RO') AND
((DBC.AccessRights.Field_5 <> 'IX') AND
((DBC.AccessRights.Field_5 <> 'RF') AND
((DBC.AccessRights.Field_5 <> 'AS') AND
((DBC.AccessRights.Field_5 <> 'SR') AND
((DBC.AccessRights.Field_5 <> 'SS') AND
((DBC.AccessRights.Field_5 <> 'MR') AND
((DBC.AccessRights.Field_5 <> 'MS') AND
(DBC.AccessRights.Field_3 =
'000000000000'XB))))))))))))))))))))))))))))))))))))))")
into Spool 3 (all_amps), which is redistributed by the rowkey
of ('00000100'XB, '00000404'XB, '000000000000'XB) to few AMPs.

```

13) We execute the following steps in parallel.

- 1) We do a single-AMP RETRIEVE step from a single partition of DBC.AccessRights by way of the primary index "Field\_1 = '00000000'XB, Field\_2 = '00000000'XB, Field\_3 = '000001000000'XB" with a residual condition of ("DBC.AccessRights.Field\_3 = '000001000000'XB") into Spool 3 (all\_amps), which is redistributed by the rowkey of ('00000404'XB, '00000404'XB, '000000000000'XB) to few AMPs.
- 2) We do an all-AMPs RETRIEVE step from DBC.AccessRights by way of an all-rows scan with a condition of ("DBC.AccessRights.Field\_8 = 'Y'") into Spool 4 (all\_amps), which is redistributed by the hash code of (DBC.AccessRights.Field\_1) to all AMPs. Then we do a SORT to

```

        order Spool 4 by row hash.
14) We do an all-AMPs JOIN step from DBC.Owners by way of a RowHash
    match scan with a condition of ("DBC.Owners.Field_2 = '00000404'XB"),
    which is joined to Spool 4 (Last Use) by way of a RowHash match
    scan. DBC.Owners and Spool 4 are joined using a merge join, with
    a join condition of ("DBC.Owners.Field_1 = Field_1"). The result
    goes into Spool 3 (all_amps), which is redistributed by the rowkey
    of ('00000404'XB, DBC.AccessRights.Field_2,
    DBC.AccessRights.Field_3) to all AMPs. Then we do a SORT to
    partition Spool 3 by rowkey.
15) We execute the following steps in parallel.
    1) We do an all-AMPs MERGE into DBC.AccessRights from Spool 3
        (Last Use).
    2) We do an INSERT into DBC.UIFDEPENDENCY.
16) We flush the DISKSPACE and AMPUSAGE caches.
17) We do an all-AMPs ABORT test from DBC.DataBaseSpace by way of the
    unique primary index "Field_1 = '00000100'XB, Field_2 =
    '000000000000'XB" with a residual condition of (
    "2.50000000000000E 005 <= (DBC.DataBaseSpace.Field_4 -
    DBC.DataBaseSpace.Field_8)").
18) We do an INSERT into DBC.DataBaseSpace.
19) We do an all-AMPs UPDATE from DBC.DataBaseSpace by way of the
    unique primary index "Field_1 = '00000100'XB, Field_2 =
    '000000000000'XB" with no residual conditions.
20) We flush the DISKSPACE and AMPUSAGE caches.
21) We spoil the parser's dictionary cache for the database.
22) Finally, we send out an END TRANSACTION step to all AMPs involved
    in processing the request.
    -> No rows are returned to the user as the result of statement 1.

```

The following is the output from a PM/API application capturing the step data of the SQL statement above.

The number is a sequential count of the rows returned from the DBS.

When a step number is repeated, it indicates that it is a parallel step. For instance, in the following example, the step number 8 is repeated multiple times. This coincides with the parallel steps found for step 8 in the above Explain.

Step text is derived information and not the true EXPLAIN text.

A sample of a PM/API application output appears as follows:

```

Submitting request MONITOR SQL; ...
Total SQL records = 1
create database testdb as perm=1e6;
=====
NumOfSteps: 33   CurStepBegNum: 33   CurStepEndNum: 33
DynamicPlan: 0   PartialSteps: 0
ZoneId: 0

```

```

SPName/SPDBName: SP NOT executing.
Default Database Name: [DBC]
33 explain steps found
Confidence= 0, RowCount      0/      4, ET      0.00/      0.00
1) First, lock [DBId=0x0406]. for exclusive.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
2) Next, we lock DBC.[TBId=0x0130] for write on a row hash.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
3) We lock DBC.DBSpace for write on a row hash.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
4) We lock DBC.Parents for write on a row hash.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
5) We lock DBC.Owners for write on a row hash.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
6) We lock DBC.AccessRights for write on a row hash.
Confidence= 0, RowCount      0/      4, ET      0.00/      0.00
7) We lock DBC.
[TBId=0x0130] for write, we lock DBC.DBSpace for write, we lock DBC.Parents for write,
we lock DBC.Owne.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
8) We do a Single-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step begins a p
arallel block .
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
8) We do a Single-AMP ABORT test from DBC.
[TBId=0x0138] by way of the unique primary index. This step is performed in .
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
8) We do a Single-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step is perform
ed in parallel.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
8) We do a Single-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step is perform
ed in parallel.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.01
8) We do an INSERT step into table DBC.DBase. This step is performed in parallel.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
8) We do a Single-
AMP UPDATE from DBC.DBase by way of the unique primary index. This step is performed i
n parallel.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
8) We do a Single-
AMP RETRIEVE step from DBC.Parents by way of the primary index into Spool 50, which is
redistributed.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
9) We do a MERGE into table DBC.Owners from Spool 50.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
10) We do an INSERT step into table DBC.Owners. This step begins a parallel block of
steps.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
10) We do a Single-
AMP RETRIEVE step from DBC.Parents by way of the primary index into Spool 51, which is
redistribute.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
11) We do a MERGE into table DBC.Parents from Spool 51.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.00
12) We do an INSERT step into table DBC.Parents. This step begins a parallel block of
steps.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.02
12) We do an INSERT step into table DBC.Accounts. This step is performed in parallel.
Confidence= 0, RowCount      0/      25, ET      0.00/      0.00
12) We do a Single-
AMP RETRIEVE step from DBC.AccessRights accessing a single partition by way of the pri
mary index in.
Confidence= 0, RowCount      0/      21, ET      0.00/      0.00
13) We do a Single-
AMP RETRIEVE step from DBC.AccessRights accessing a single partition by way of the pri
mary index in.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00

```

```

13) We do an All-AMPs RETRIEVE step from DBC.AccessRights by way of an all-
rows scan into Spool 53, which is redistrib.
Confidence= 0, RowCount      0/      46, ET      0.00/      0.01
14) We do an All-AMPs JOIN step from DBC.Owners by way of an all-
rows scan, which is joined to Spool 53. table Owners .
Confidence= 0, RowCount      0/      46, ET      0.00/      0.01
15) We do a MERGE into table DBC.AccessRights from Spool 52. This step begins a paral
lel block of steps.
Confidence= 0, RowCount      0/      1, ET      0.00/      0.02
15) We do an INSERT step into table [TBId=0x0130]. This step ends a parallel block of
steps.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
16) We flush the DISKSPACE and AMPUSAGE caches.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.00
17) We do an All-AMPs ABORT test from DBC.DBSpace by way of the unique primary index.
Confidence= 0, RowCount      0/      4, ET      0.00/      0.00
18) We do an INSERT step into table DBC.DBSpace.
Confidence= 0, RowCount      0/      4, ET      0.00/      0.00
19) We do an All-AMPs UPDATE from DBC.DBSpace by way of the unique primary index.
Confidence= 0, RowCount      0/      0, ET      0.00/      0.01
20) We flush the DISKSPACE and AMPUSAGE caches.
Confidence= 0, RowCount      0/      -1, ET      0.00/      -1.00
21) We Spoil the parser's dictionary cache for the database.
Confidence= 0, RowCount      0/      -1, ET      0.00/      0.00
22) We send out an END TRANSACTION step to all AMPs involved in processing the request.

```

## MONITOR VERSION

Identifies the highest version number (MonVerId) the database monitor function supports, which determines the requests and data fields available for use.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data.</p> <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 3 or later.                                                                                                                                                                                                                                                                                                                                                                                                                         |

### Usage Notes - MONITOR VERSION

The MONITOR VERSION request does not check for access privileges.

## CLv2 Response Parcels

The MONITOR VERSION request is treated internally as a one-statement request that generates one response. The statement response returned from the database contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                | Comments/Key Parcel Body Fields                                                                                                                                                              |
|-----------------|---------------|---------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                     | StatementNo = 1<br>ActivityCount = Highest supported version<br>ActivityType = 108 (PCLMONVER)                                                                                               |
| DataInfo        | 71            | 6 to 64100                                                                                                    | Optional; this parcel is present if request was IndicData parcel.                                                                                                                            |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100(record mode)</li> <li>6 to 64100(indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. This record contains a 2-byte bitmap field indicating the supported functions and the current version number. |
| EndStatement    | 11            | 6                                                                                                             | StatementNo = 2-byte integer                                                                                                                                                                 |
| EndRequest      | 12            | 4                                                                                                             | None                                                                                                                                                                                         |

For more information on parcel body fields, see *Teradata® Call-Level Interface Version 2 Reference for Mainframe-Attached Systems*, B035-2417 or *Teradata® Call-Level Interface Version 2 Reference for Workstation-Attached Systems*, B035-2418.

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The statement response results in a Record parcel containing:

| Field/Column Name | Data Type            | Description                                                                                                                                                                          |
|-------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FunctionBitmap    | SMALLINT<br>NOT NULL | Functions are: <ul style="list-style-type: none"> <li>Bit0 = ModifyAccount</li> <li>Bit1 = Monitor SQL</li> <li>Bit2 = 15 - Set to zero (This bit is not currently used.)</li> </ul> |

| Field/<br>Column Name | Data Type            | Description                                                                                                                                                                                                                                                                 |
|-----------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |                      | <b>Note:</b><br>The bits are 1 if supported and 0 if not.                                                                                                                                                                                                                   |
| CurrentVersion        | SMALLINT<br>NOT NULL | Current monitor version number.<br><br><b>Note:</b><br>If you are using MONITOR software version ID ( <i>mon_ver_id</i> ) 8 or earlier, this output parameter is not available. For more information, see <a href="#">Impact of Object Name Length on PM/API Requests</a> . |

### Sample Input - CLv2 Request

This example shows how the parcels for a MONITOR VERSION request, built by CLv2, appear when sent to the database server. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body       |                 |
|--------|------|--------|------------|-----------------|
| Num    | Name | Bytes  | Field      | Value           |
| 0001   | Req  | 19     | Request    | MONITOR VERSION |
| 0003   | Data | 8      | MonVerID   | 12              |
| 0004   | Resp | 6      | BufferSize | 64000           |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This example shows the values returned in character text format for the MONITOR VERSION request. Your application program may display returned values in a different format.

```
SUCCESS parcel:
StatementNo=1,    ActivityCount=1,
ActivityType=111, FieldCount=2
MONITOR VERSION:
Function bitmap: 3, Current monitor version=12
```



## MONITOR VIRTUAL CONFIG

Collects information on virtual processor (vproc) availability.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using the indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data.</p> <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This can be version 2 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                                                                                                                                                                                                    |

### Monitor Privileges

To use this request, you must have any one of the following monitor privileges as part of your default role or any of these privileges must be granted directly to you:

- ABORTSESSION
- MONRESOURCE
- MONSESSION
- SETRESRATE
- SETSESSRATE

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

### Usage Notes - MONITOR VIRTUAL CONFIG

Information regarding vproc status is returned for all AMP, PE, and TVS vprocs in the system.

MONITOR VIRTUAL CONFIG is most useful when used with the MONITOR VIRTUAL SUMMARY request for doing a quick overall system health check. For more information, see [Relationship Between MONITOR VIRTUAL CONFIG and MONITOR VIRTUAL SUMMARY](#).

If you use MONITOR VIRTUAL CONFIG, you do not need to dump out the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view) to see if there is a physical problem with the system.

## CLiv2 Response Parcels

The MONITOR VIRTUAL CONFIG request is treated internally as a two statement request with each statement generating a response. The two statement response returned from the database contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                 |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = 91 (PCLMONVCONFIG)                                                                                                       |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional; this parcel is present if request was IndicData parcel.                                                                                                               |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the BYNET status data and the type of system running the database software. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                    |
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 2<br>ActivityCount = Number of vprocs<br>ActivityType = 91 (PCLMONVCONFIG)                                                                                        |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional; this parcel is present if request was IndicData parcel.                                                                                                               |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. This record contains the vproc-specific information; one record per vproc.                       |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                    |
| EndRequest      | 12            | 4                                                                                                               | None                                                                                                                                                                            |

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Statement 1

The following table describes the order in which the Record parcel in the first statement of the MONITOR VIRTUAL CONFIG returns the BYNET status values and the system type.

| Column | Field/<br>Column<br>Name | Data Type               | Description                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|--------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | NetAUp                   | VARCHAR (1)<br>NOT NULL | <p>Status of the BYNETs (if there are more than two, the first two) on a system-wide basis.</p> <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <p><b>Note:</b><br/>This output parameter is available on monitor software version 9 or later only.</p> |
| 2      | NetBUp                   | VARCHAR (1)<br>NOT NULL | <p>Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:</p> <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> <p><b>Note:</b><br/>This output parameter is available on monitor software version 9 or later only.</p> |
| 3      | SystemType               | VARCHAR (7)<br>NOT NULL | <p>Type of system running the database software, such as 5650, 6700, or 'Other'.</p> <p>If all the nodes in the system are the same type, this field returns the type of the system.</p> <p>If any of the nodes are of a different type, this field returns 'Mixed'.</p> <p><b>Note:</b><br/>This output parameter is available on monitor software version 9 or later only.</p>                                                    |

## Statement 2

The Record parcels in the second statement of the MONITOR VIRTUAL CONFIG response return one record for each processor, with six fields in each record. Records are sorted based on VProcNo. For example, if you have 32 vprocs, 32 records are returned with specific information for each vproc, in addition to the one record of BYNET data for the whole system.

| Field/<br>Column<br>Name | Data Type               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Procid                   | INTEGER                 | ID associated with a node.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.                                                                                                                                                                                                                                                                                                                                        |
| VProcNo                  | SMALLINT<br>NOT NULL    | ID of an AMP (that is, a set of disk) and the associated tasks or processes that, in combination, make up the AMP), PE or TVS vproc.                                                                                                                                                                                                                                                                                                                                                                                                                  |
| VProcType                | VARCHAR (3)<br>NOT NULL | Type of vproc: <ul style="list-style-type: none"> <li>• AMP</li> <li>• PE</li> <li>• MISC</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| HostId                   | SMALLINT                | Logical host ID for the PEs. This field shows NULL for the AMP or TVS vprocs associated with this record. Each channel- or TCP/IP network-connected host is assigned an ID at the time the system is configured. Each PE is assigned to a (single) channel-connected host (or client) or a TCP/IP network-connected host. The host ID of zero is reserved for the PEs processing internal sessions.                                                                                                                                                   |
| Status                   | VARCHAR (1)<br>NOT NULL | Status of the vproc associated with this record. A vproc is considered up or down from the standpoint of whether the vproc is helping a query process SQL statements. For example, an AMP doing offline recovery is considered to be down because the AMP is not helping to process SQL statements. On the other hand, an up vproc is one that is online and fully up or is in online recovery.<br>The status of the vproc: <ul style="list-style-type: none"> <li>• U = The vproc is up/online.</li> <li>• D = The vproc is down/offline.</li> </ul> |
| DiskSlice                | SMALLINT                | Virtual disk ID defining the portion of a physical disk assigned to an AMP. This value is NULL for TVS vprocs.                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## Sample Input - CLlv2 Request

This example shows how the parcels for a MONITOR VIRTUAL CONFIG request, built by CLlv2, appear when sent to the database server. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body       |                        |
|--------|------|--------|------------|------------------------|
| Num    | Name | Bytes  | Field      | Value                  |
| 0001   | Req  | 26     | Request    | MONITOR VIRTUAL CONFIG |
| 0003   | Data | 6      | MonVerID   | 2                      |
| 0004   | Resp | 6      | BufferSize | 64000                  |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This example shows the values returned in character text format for the MONITOR VIRTUAL CONFIG request. Your application program may display returned values in a different format.

```
Submitting request MONITOR VIRTUAL CONFIG; ...
```

```
NetAUp:  U NetBUp:  U
SystemType: 5500C
```

```
8 vproc(s) found
```

| ProcId | Node Loc  | VProcNo | VProcType | HostId |
|--------|-----------|---------|-----------|--------|
| Status | DiskSlice |         |           |        |
| =====  | =====     | =====   | =====     | =====  |
| =====  | =====     |         |           |        |
| 10001  | (1-1)     | 0       | AMP       | 0 U 0  |
| 10001  | (1-1)     | 1       | AMP       | 0 U 1  |
| 10001  | (1-1)     | 2       | AMP       | 0 U 2  |
| 10001  | (1-1)     | 3       | AMP       | 0 U 3  |
| 10001  | (1-1)     | 28670   | TVS       | 0 U 0  |
| 10001  | (1-1)     | 28671   | TVS       | 0 U 0  |
| 10001  | (1-1)     | 30718   | PE        | 1 U 0  |
| 10001  | (1-1)     | 30719   | PE        | 1 U 0  |

### Relationship Between MONITOR VIRTUAL CONFIG and MONITOR VIRTUAL SUMMARY

Use MONITOR VIRTUAL SUMMARY with the MONITOR VIRTUAL CONFIG request for an overall system status. These are low overhead requests.

- Execute the MONITOR VIRTUAL SUMMARY request every 5 or 10 minutes for a low-cost, continuous monitoring of your system.
- Execute the MONITOR VIRTUAL CONFIG request to get a picture of your system configuration at defined times, such as at the beginning of a day, various times during the day, or when the system is down.

Using these requests to spot problems, such as abnormal AMP CPU load balancing, and possible sources of system performance bottlenecks. For example, if the HiCPUAMPUse, HiCPUPEUse, LoCPUAMPUse, and LoCPUPEUse figures are consistently widely separated and do not approximate the AMPAvgCPU figure, you may need to evaluate whether the system is using available resources efficiently. Alternatively, if the PEAvgCPU is consistently much higher than the AMPAvgCPU, the system may not be configured to efficiently use AMP resources. How often you perform a health check of your system depends on the size of your system and the type of applications run.

Knowledge of the overall system status can help you to determine:

| Concern                                                     | Comments                                                                                                                                                                                                        |
|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| When to run production applications, especially large ones  | For example, if you have a down AMP, you may decide that it is less costly to recover the AMP first and run the job than to run the job without full system availability.                                       |
| Why an application runs more slowly than usual              | This situation may be caused by a down AMP, which results in the backup AMP doing more work, specifically, doing backup and primary processing. This, in turn, could cause your application to run more slowly. |
| Whether all vprocs have come back up after a system restart | Examine the Status value returned in a MONITOR VIRTUAL CONFIG request to determine whether each vproc is up or down (see the MONITOR VIRTUAL CONFIG Status field).                                              |

The response data returned by MONITOR VIRTUAL CONFIG is similar in content to the response data returned by a MONITOR VIRTUAL RESOURCE request, but it is in an abbreviated form. In your initial problem analysis, if the information returned from a MONITOR VIRTUAL SUMMARY query does not give you enough data. For example, you need BYNET or CPU% busy information and you may want to use MONITOR VIRTUAL RESOURCE to obtain more detailed resource usage data.

## MONITOR VIRTUAL RESOURCE

Collects performance information for each AMP, PE, or TVS vproc and returns:

- System-wide (identical for all vprocs) data
- vproc-specific data

### Input Data

| Element        | Data Type | Description                                                                                                                                                        |
|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i> | BYTE      | Indicator bits that specify which fields to treat as NULL if you are using the indicator mode.<br>Each bit in the byte corresponds to one field in the input data. |

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                |
|-------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |                      | <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This can be version 2 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                      |

## Monitor Privileges

To use this request, you must have the MONRESOURCE privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR VIRTUAL RESOURCE

You can use the MONITOR VIRTUAL RESOURCE request to:

- Expand on the data reported by the MONITOR VIRTUAL SUMMARY request.
 

In your initial problem analysis, a MONITOR VIRTUAL SUMMARY request may indicate a performance or system problem. MONITOR VIRTUAL RESOURCE allows you to collect RSS data on a vproc by vproc basis.
- Continually monitor your system.
 

Monitor your system continually on a periodic basis, for example, every 10 minutes. Use this request to build a normal baseline profile for your system. When you notice something abnormal, such as the last reading is significantly different from the normal baseline reading, or when a user complains that a job is slow, this request can tell you if there is a parallel efficiency problem or a constraint to throughput, and which vproc is causing it. For example, if one PE shows a much higher usage than the other PEs, that PE might be overloaded. Also, if one AMP is loaded more heavily than the other AMPs, you may have problems with a skewed index.
- Determine whether a new application can be added to the current system load without disruption.
 

The vproc usage information collected by this request can help you evaluate the impact of adding new applications to an already heavily utilized system and help you plan potential system upgrades.
- Help resolve problems that session-level usage information cannot resolve.

When the MONITOR SESSION request does not show any cause for the problem, this request can supply information regarding congestion, memory allocations, BYNET outages, and system status.

The MONITOR VIRTUAL RESOURCE request can provide information about:

- How the system is being used (for example, the number of sessions associated with each vproc and the percentage of CPU usage by vproc).
- How system resource usage is spread across the vprocs (for example, whether the vprocs are used evenly).
- How much physical disk I/O, BYNET traffic, or host reads and writes are occurring.
- Whether congestion or excessive swapping is a problem on any vproc or group of vprocs.

The MONITOR VIRTUAL RESOURCE request returns some of the same fields found in the resource usage tables. You can use both MONITOR VIRTUAL RESOURCE and resource usage data for problem detection. Unlike resource usage data, MONITOR VIRTUAL RESOURCE data is near real time, and requires less overhead to produce, but is less comprehensive. MONITOR VIRTUAL RESOURCE data can help detect:

- Poor AMP CPU parallel efficiency
- Poor disk parallel efficiency
- A higher than expected disk read/write ratio
- A high swap I/O rate

If MONITOR VIRTUAL RESOURCE does not give you all the detailed data you need for problem detection, run one or more of the resource usage macros for the AMP, PE, or TVS vprocs. See *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099 for more information on problem detection and the resource usage macros.

---

**Note:**

You must set the rate by which vproc resource usage data is updated in memory (ResMonitor rate) to nonzero for the MONITOR VIRTUAL RESOURCE request to return meaningful data. If you set the ResMonitor rate to zero, NULL is returned for all vproc usage data.

---

Also note that if the TVS vproc is configured on a node without any AMPs, all fields will return a value of zero.

After a system outage or a change in the ResMonitor rate, do not request data again until after completion of the first collection period requested after the crash or change in rate. Otherwise, the data returned will contain NULL for all columns **except** NetAU, NetBU, SampleSec, CollectionDate, CollectionTime, VProcType, ProclD, VProcNo, HostID/ClusterNo, and Status, and may not be fully representative. This is because after a system failure, the in-memory counters are reset, and typically the contents of the counters are not well defined until a full collection period has elapsed. In fact, if you were logged on prior to the system outage and you issue your first MONITOR VIRTUAL RESOURCE request after the outage, you will receive a warning that the database system has been restarted.



## CLlv2 Response Parcels

The MONITOR VIRTUAL RESOURCE request is treated internally as a two statement request, with each statement generating a response. The two statement response returned from the database contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                                                                                            |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = 95 (PCLMONVRES)                                                                                                                                                                                     |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional; this parcel is present if request was IndicData parcel.                                                                                                                                                                                          |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. One record is returned that contains the duration of the collection period (in seconds), BYNET status, and the date and time the Virtual Resource cache was last refreshed. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                                                                                               |
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 2<br>ActivityCount = Number of vprocs<br>ActivityType = 95 (PCLMONVRES)                                                                                                                                                                      |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional; this parcel is present if request was IndicData parcel.                                                                                                                                                                                          |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. One record per vproc is returned that contains a description for each vproc in the system.                                                                                  |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                                                                                               |
| EndRequest      | 12            | 4                                                                                                               | None                                                                                                                                                                                                                                                       |

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

**Statement 1**

The Record parcel in the first statement of the MONITOR VIRTUAL RESOURCE response returns global data about collection duration and BYNET status in the order listed below.

| Field/Column Name | Data Type               | Description                                                                                                                                                                                                                                                                                                          |
|-------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NetAUp            | VARCHAR (1)<br>NOT NULL | Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> |
| NetBUp            | VARCHAR (1)<br>NOT NULL | Status of the BYNETs (if there are more than two, the first two) on a system-wide basis: <ul style="list-style-type: none"> <li>• U = All node BYNETs are up/online.</li> <li>• D = One or more node BYNETs is down/offline.</li> <li>• "" = A temporary condition where the BYNET data is not available.</li> </ul> |
| SampleSec         | SMALLINT<br>NOT NULL    | Duration of the collection period in seconds. This field returns the ResMonitor rate. See <a href="#">Data Collection</a> and <a href="#">SET RESOURCE RATE</a> for more information on ResMonitor.                                                                                                                  |
| CollectionDate    | DATE<br>NOT NULL        | Date the Virtual Resource cache was last refreshed.                                                                                                                                                                                                                                                                  |
| CollectionTime    | FLOAT<br>NOT NULL       | Time the Virtual Resource cache was last refreshed.                                                                                                                                                                                                                                                                  |

**Statement 2**

The response to the second statement results in multiple Record parcels that consist of a record for each vproc in the system. The Record parcels in the second statement of the MONITOR VIRTUAL RESOURCE response can return multiple records, specifically, one record of 32 fields for each vproc in the system. For example, if you have 50 vprocs, 50 records are returned with specific information for each vproc, one record describing the collection period, and BYNET status for the entire system. Records are sorted by VProcType and VProcNo.

The following table shows the order in which the data is returned from the Record parcel.

| Column Name | Data Type               | Description                                                                                          |
|-------------|-------------------------|------------------------------------------------------------------------------------------------------|
| VprocType   | VARCHAR (3)<br>NOT NULL | Type of vproc: <ul style="list-style-type: none"> <li>• AMP</li> <li>• PE</li> <li>• MISC</li> </ul> |
| ProcId      | INTEGER                 | ID associated with a node.                                                                           |

| Column Name      | Data Type                  | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  |                            | This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.                                                                                                                                                                                                                                                                                                                                                                                             |
| VprocNo          | SMALLINT<br>NOT NULL       | ID of an AMP (that is, a set of disks and the associated tasks or processes that, in combination, make up the AMP), PE or TVS vproc.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| HostId/ClusterNo | SMALLINT                   | For a PE vproc, this field, HostId, identifies one of the hosts or LANs associated with the described PE.<br>For an AMP vproc, this field, ClusterNo, identifies the cluster to which this AMP is assigned.<br><br><b>Note:</b><br>This field is not applicable to TVS vproc and returns NULL.                                                                                                                                                                                                                                                                               |
| CPUUse           | FLOAT<br>range 0 - 100%    | % of CPU usage not spent being idle.<br>The value is computed from ResUsageSvpr table data as, where <i>NCPUs</i> is the number of CPUs in the node:<br>$100.00 * (CPUUExecPart00 + CPUUExecPart01 + \dots + CPUUExecPart47 + CPUUServPart00 + CPUUServPart01 + \dots + CPUUServPart47) / (NCPUs * SampleSec * 100)$<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a> .                                                                                                                                         |
| Status           | VARCHAR<br>(1)<br>NOT NULL | Status of the node, AMP, PE, or TVS vproc associated with this record: <ul style="list-style-type: none"> <li>• U = Up/online.</li> <li>• D = Down/offline.</li> </ul> A node is up (U) when it is: <ul style="list-style-type: none"> <li>• Configured into the system</li> <li>• Online</li> <li>• Capable of actively performing tasks associated with normal database activity</li> </ul> Down (D) represents all other potential states.                                                                                                                                |
| SessLogCount     | SMALLINT                   | Number of current sessions logged to this PE. A logged on session is either a session whose logon request was delivered to this PE, or a session that was switched to this PE following its logon.<br><br><b>Note:</b><br>The SessLogCount field contains the SubPoolId if the vproc type is TVS.<br>SubpoolId identifies the subpool associated with the allocator vproc. A subpool defines a set of storage and allocator vprocs assigned to that storage.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a> . |
| SessRunCount     | SMALLINT                   | Number of current sessions whose Initiate Requests (TSR messages) are addressed to this vproc. For example: <ul style="list-style-type: none"> <li>• PEs have a SessRunCount that includes all the Teradata SQL and MONITOR sessions logged on to that PE.</li> </ul>                                                                                                                                                                                                                                                                                                        |

| Column Name   | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               |           | <ul style="list-style-type: none"> <li>AMPs may have a nonzero SessRunCount, since AMPs receive TSR messages from FastLoad or MultiLoad logons.</li> </ul> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> <p><b>Note:</b><br/>This field is not applicable to TVS vprocs.</p>                                                                                                                                                                                                                                                                                  |
| DiskUse       | FLOAT     | <p>% of disk usage per AMP.<br/>This value is computed from the ResUsageSvdsk table data:<br/>OutReqTime / SampleSec</p> <p><b>Note:</b><br/>DiskUse does not take into account overlapping of operations among multiple storage controllers, but it allows for the possibility of multiple requests for the same controller.</p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                                                                                                                                               |
| DiskReads     | FLOAT     | <p>Total number of physical disk reads per AMP during the collection period.<br/>This value is computed from the ResUsageSvpr table data as:<br/>FilePCiAcqReads + FilePDbAcqReads + FileSCiAcqReads + FileSDbAcqReads + FileTjtAcqReads + FileAPtAcqReads;<br/>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> <p><b>Note:</b><br/>This field is not applicable to TVS and PE vprocs.</p>                                                                                                                                                                         |
| DiskWrites    | FLOAT     | <p>Total number of physical disk writes per AMP during the collection period.<br/>For PE and AMP-level displays, this value is computed from ResUsageSvpr table data as:<br/>FilePCiFWrites + FilePDbFWrites + FileSCiFWrites + FileSdbFWrites + FileTjtFWrites + FileAPtFWrites + FilePCiDyaWrites + FilePDbDyaWrites + FileSciDyaWrites + FileSdbDyaWrites + FileTjtDyaWrites + FileAptDyaWrites<br/>For TVS vproc displays, this value is computed from ResUsageSvpr table data as:<br/>AllocatorMapIOsDone<br/>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> |
| DiskOutReqAvg | FLOAT     | <p>Average number of outstanding disk requests.<br/>For AMP-level displays, this value is computed from ResUsageSvdsk table data, assuming <math>n</math> is the number of storage devices used by this vproc:<br/>(ReadRespTot 1 + WriteRespTot 1 + ... + ReadRespTot <math>n</math> + WriteRespTot <math>n</math>) / CentiSecs</p>                                                                                                                                                                                                                                                                                               |

| Column Name     | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |           | <p><b>Note:</b><br/>           This field is not applicable to PE vprocs.<br/>           For TVS vproc-level displays, this value is computed from ResUsageSvpr table data as:<br/>           AllocatorMapIOsStarted - AllocatorMapIOsDone</p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                                                           |
| HostBlockReads  | FLOAT     | <p>Number of message blocks (one or more messages sent in one physical group) received from all clients.<br/>           This value corresponds to the column totals in the ResUsageShst table supplying HostBlockReads for this vproc.<br/>           This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> <p><b>Note:</b><br/>           This field is not applicable to AMP or TVS vprocs.</p> |
| HostBlockWrites | FLOAT     | <p>Number of message blocks (that is, one or more messages sent in one physical group) sent to all hosts.<br/>           This value corresponds to the column totals in the HstBlkWrts column of the ResUsageShst table.</p> <p><b>Note:</b><br/>           This field is not applicable to AMP or TVS vprocs.</p>                                                                                                                                          |
| MemAllocates    | FLOAT     | <p>Number of segments allocated to memory resources.<br/>           This value is calculated from the following ResUsageSvpr table column:<br/>           MemCtxtAllocs<br/>           This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                                                                                                                      |
| MemAllocateKB   | FLOAT     | <p>Value represents the change in vproc-level memory. MemAllocateKB represents a delta from the previous reporting period. It reports negative values as less memory is used.<br/>           This value is calculated from the following ResUsageSvpr column:<br/>           MemCtxtAllocs * FIXEDPAGE_SIZE<br/>           This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                  |
| PercentService  | FLOAT     | <p>% of CPU resources spent in PDE user service processing.<br/>           This value is computed from the ResUsageSvpr table data, where x represents the number of CPUs:<br/> <math>(CPUUServPart00 + CPUUServPart01 + \dots + CPUUServPart47) / (x * SampleSec)</math><br/>           This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                    |

| Column Name      | Data Type               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------------------|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PercntAMPWT      | FLOAT<br>range 0 - 100% | <p>% of CPU resources used by either the AMP Worker Task (Partition 11) or by the TVS Task (Partition 31) depending on the type of vproc this record represents. For information on partition assignments, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</p> <p>This value depends on the number of CPUs in the node but does not exceed 100%. It is computed from the ResUsageSvpr table data, where x represents the number of CPUs on a node:</p> <p>For AMP vprocs:<br/> <math display="block">(\text{CPUUExecPart11}) / (x * \text{SampleSec})</math> </p> <p>For TVS vprocs:<br/> <math display="block">(\text{CPUUExecPart31}) / (x * \text{SampleSec})</math> </p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> |
| PercntParser     | FLOAT<br>range 0 - 100% | <p><b>Note:</b><br/>This field is obsolete and returns zero or NULL.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| PercntDispatcher | FLOAT<br>range 0 - 100% | <p>% of CPU resources spent in PE Dispatcher processing.</p> <p>This value depends on the number of CPUs in the node but does not exceed 100%. This value is computed from the ResUsageSvpr table data, where x represents the number of CPUs on a node:<br/> <math display="block">(\text{CPUUExecPart13} + \text{CPUUServPart13}) / (x * \text{SampleSec})</math> </p> <p><b>Note:</b><br/>The PercntParser CPU time is included in the PercntDispatcher value.</p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p> <p><b>Note:</b><br/>PercntDispatcher is not applicable to AMP and TVS vprocs.</p>                                                                                                                                                  |
| NetReads         | FLOAT                   | <p>Number of Reads from the BYNET to the vproc.</p> <p>This value is computed from the ResUsageSvpr table data as follows:<br/> <math display="block">\text{NetBrdReads} + \text{NetPtPReads}</math> </p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| NetWrites        | FLOAT                   | <p>Number of messages written from the AMP, PE, or vproc to the BYNET during the collection period.</p> <p>This value is computed from the ResUsageSvpr table data as follows:<br/> <math display="block">\text{NetBrdWrites} + \text{NetPtPWrites}</math> </p> <p>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| MaxIOResp        | FLOAT                   | <p>Value is computed from ResUsageSvpr table data using the IoRespMax field. IoRespMax is the maximum I/O response time in milliseconds on an AMP. For details, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

| Column Name | Data Type | Description                                                                                                                                                                            |
|-------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |           | <b>Note:</b><br>This field is not applicable to PE and TVS vprocs.<br><br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL RESOURCE</a> . |

### Sample Input - CLIV2 Request

This example shows how the parcels for a MONITOR VIRTUAL RESOURCE request, built by CLIV2, look when sent to the database server. The size of the response buffer is set in the example at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body       |                          |
|--------|------|--------|------------|--------------------------|
| Num    | Name | Bytes  | Field      | Char/Decimal             |
| 0001   | Req  | 28     | Request    | MONITOR VIRTUAL RESOURCE |
| 0003   | Data | 6      | MonVerID   | 9                        |
| 0004   | Resp | 6      | BufferSize | 64000                    |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This example shows the values returned in character text format (a record for each vproc) for the MONITOR VIRTUAL RESOURCE request. Your application program may display returned values in a different format.

#### Note:

You can rename the SampleSec field in your application. In the output below, the SampleRate value is the SampleSec value.

Pay attention to SampleRate when interpreting the results of this request.

```
Submitting request MONITOR VIRTUAL RESOURCE; ...
```

```
NetAUp:  U NetBUp:  U
```

```
SampleRate: 60
```

```
Collection Date/Time: 07/06/2016 13:48:00.00
```

```

VprocNo:      0   Vproctype: AMP   Status:  U
ProcId:      10001 (1-1) HostId/ClusterNo: 0

SessLogCount: 0   SessRunCount: 0

CPUUse:       24.7   PctService:  0.4
PctAMPWT:     24.3   DiskUse: 12.7
DiskReads:    5.00   DiskWrites: 1323.00   DiskOutReqAvg:  0.17

NetReads:     74.00   NetWrites:   69.00
NVMemAllocSegs: 265.00

```

```

-----
VprocNo:      1   Vproctype: AMP   Status:  U
ProcId:      10001 (1-1) HostId/ClusterNo: 1

SessLogCount: 0   SessRunCount: 0

CPUUse:       24.7   PctService:  0.4
PctAMPWT:     24.2   DiskUse: 12.9
DiskReads:    7.00   DiskWrites: 1345.00   DiskOutReqAvg:  0.17

NetReads:     55.00   NetWrites:   56.00
NVMemAllocSegs: 300.00

```

```

-----
VprocNo:      2   Vproctype: AMP   Status:  U
ProcId:      10001 (1-1) HostId/ClusterNo: 0

SessLogCount: 0   SessRunCount: 0

CPUUse:       24.8   PctService:  0.6
PctAMPWT:     24.2   DiskUse: 13.6
DiskReads:    6.00   DiskWrites: 1355.00   DiskOutReqAvg:  0.19

NetReads:     54.00   NetWrites:   57.00
NVMemAllocSegs: 302.00

```

```

-----
VprocNo:      3   Vproctype: AMP   Status:  U
ProcId:      10001 (1-1) HostId/ClusterNo: 1

SessLogCount: 0   SessRunCount: 0

```



```

CPUUse:      24.6    PctService:  0.4
PctAMPWT:    24.2    DiskUse: 13.4
DiskReads:   10.00   DiskWrites: 1355.00   DiskOutReqAvg:  0.19

```

```

NetReads:    55.00   NetWrites:   60.00
NVMemAllocSegs: 319.00

```

```

-----
VprocNo: 28670  Vproctype: TVS   Status:  U
ProcId:   10001 (1-1) HostId/ClusterNo: 0

```

```
SubPoolId: 1
```

```

CPUUse:      0.1    PctService:  0.0
CPUExecPart31: 0.1      DiskUse:  0.0
AllocatorMapIOsDone: 94.00   PendingAllocatorMapIOs:  0.00

```

```

NetReads:    99.00   NetWrites:  100.00
NVMemAllocSegs:  0.00

```

```

-----
VprocNo: 28671  Vproctype: TVS   Status:  U
ProcId:   10001 (1-1) HostId/ClusterNo: 0

```

```
SubPoolId: 0
```

```

CPUUse:      0.1    PctService:  0.0
CPUExecPart31: 0.1      DiskUse:  0.0
AllocatorMapIOsDone: 93.00   PendingAllocatorMapIOs:  0.00

```

```

NetReads:    98.00   NetWrites:   97.00
NVMemAllocSegs:  0.00

```

```

-----
VprocNo: 30718  Vproctype: PE    Status:  U
ProcId:   10001 (1-1) HostId/ClusterNo: 1025

```

```
SessLogCount: 0   SessRunCount: 0
```

```

CPUUse:      0.0    PctService:  0.0
PctParser:    0.0    PctDispatcher: 0.0   HstBlkRds:  0.00   HstBlkWrts:
0.00

```

```

NetReads:   100.00   NetWrites:    99.00
NVMemAllocSegs:    0.00

-----
VprocNo:   30719   Vproctype: PE   Status:  U
ProcId:    10001 (1-1) HostId/ClusterNo: 1025

SessLogCount: 1   SessRunCount: 0

CPUUse:      0.0   PctService:  0.0
PctParser:   0.0   PctDispatcher: 0.0   HstBlkRds:    0.00   HstBlkWrts:
0.00

NetReads:    18.00   NetWrites:    17.00
NVMemAllocSegs:    0.00

```

## Warning and Error Messages

All users who are logged on and issue a MONITOR VIRTUAL RESOURCE request after a system restart, or after the last rate change can expect to receive a warning message. Generally, two types of situations can produce warning messages:

- After a system restart, before and after a collection period has expired.  
If the collection period has not expired and the user issues the next MONITOR VIRTUAL RESOURCE request, many of the values returned are NULL.
- After the last rate change, before and after a collection period has expired.  
If the collection period has not expired and the user issues the next MONITOR VIRTUAL RESOURCE request, many of the values returned are NULL.

For a discussion of general warning and error messages that may be returned by MONITOR VIRTUAL RESOURCE and other requests, see [Common Warning and Error Messages](#).

For more detailed information on warning and error messages, see *Teradata Vantage™ - Database Messages*, B035-1096.

## Relationship Between MONITOR VIRTUAL RESOURCE and ABORT SESSION

If you executed an ABORT SESSION request, data returned in a MONITOR PHYSICAL RESOURCE or MONITOR VIRTUAL RESOURCE request may be altered. Whether you notice the change in data depends on the scope of the ABORT SESSION request. For example, if you execute an ABORT SESSION and log off all of the sessions associated with a specific host (or client), the PEs associated with that client will report a large decrease in resource consumption. However, if the ABORT SESSION request only aborts one transaction from one session, you may not notice a change in AMP or PE resource use.

## Relationship Between MONITOR VIRTUAL RESOURCE and SET RESOURCE RATE

You must execute the SET RESOURCE RATE request to activate resource data collection before you execute a MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request. This means that you must set the resource monitoring rate (ResMonitor) to nonzero. If the ResMonitor rate is set to zero, you will receive an error message.

A change in the resource collection rate by User A, for example, may affect the data reported by MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request made by User B. If the ResMonitor rate is altered, User B receives a warning message when executing a subsequent MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request.

## MONITOR VIRTUAL SUMMARY

Collects global summary information on system utilization.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode. |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 2 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |

### Monitor Privileges

To use this request, you must have any one of the following monitor privileges as part of your default role or any of these privileges must be granted directly to you:

- ABORTSESSION
- MONRESOURCE
- MONSESSION
- SETRESRATE
- SETSESSRATE

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - MONITOR VIRTUAL SUMMARY

The MONITOR VIRTUAL SUMMARY request reports the following types of information:

- CPU usage (average by vproc type online only)
- Disk usage (average, high, and low by Vproc ID)
- AMP usage (average, high, and low by AMP ID)
- PE usage (average, high, and low by PE ID)
- Number of logged on sessions
- Rate information:
  - Vproc resource logging rate
  - Vproc resource monitoring rate
  - Session-level system
  - Local monitoring rates
- Current software release and version numbers

You must set the ResMonitor rate (the rate at which vproc resource usage data is updated in memory) to a nonzero value for the MONITOR VIRTUAL SUMMARY request to return meaningful data. If you set the ResMonitor rate to zero, NULL is returned for all columns related to vproc utilization.

After a system outage or a change in the ResMonitor rate, do not request data again until after the completion of the first collection period requested after the outage or change in rate. If you ignore this, the data returned will contain NULL. This is because after an outage, the in-memory counters are reset, and usually the contents of the counters are not well defined until a full collection period has elapsed. For example, if you were logged on prior to the system outage and you issue your first MONITOR VIRTUAL SUMMARY request after the outage, you will receive a warning that the database server has been restarted.

## CLIV2 Response Parcels

The response returned from the database resembles a summary of the type of response returned by a MONITOR VIRTUAL RESOURCE request. The response is one row of 35 fields.

The response returned from the database contains the following sequence of parcel types.

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                         |
|-----------------|---------------|----------------|---------------------------------------------------------|
| Success         | 8             | 18 to 273      | ActivityCount = 1<br>ActivityType = 93 (PCLMONVSUMMARY) |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                                             |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional; this parcel is present if request was IndicData parcel.                                                                                                                                           |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on request (Data or IndicData), data is in record or indicator mode. This contains the IndicData virtual summary information and the date and time the Virtual Resource cache was last refreshed. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                                                |
| EndRequest      | 12            | 4                                                                                                               | None                                                                                                                                                                                                        |

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The statement response results in a Record parcel containing resource usage information.

| Field/<br>Column Name | Data Type                | Description                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AMPAvgCPU             | FLOAT<br>range 0 to 100% | Average % CPU usage (CPUUse) of all online AMPs in the configuration.<br>Assuming $n$ is the number of online AMPs in the configuration, AMPAvgCPU is computed from CPUUse data as:<br>$(CPUUse_1 + \dots + CPUUse_n) / n$<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .              |
| AMPAvgDisk            | FLOAT                    | Average physical disk usage (DiskUse) of all online AMPs in the configuration.<br>Assuming $n$ is the number of online AMPs in the configuration, AMPAvgDisk is computed from DiskUse data as:<br>$(DiskUse_1 + \dots + DiskUse_n) / n$<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> . |
| AMPAvgDiskIO          | FLOAT                    | Average physical disk DiskReads and DiskWrites of all online AMPs in the configuration.<br>Assuming $n$ is the number of online AMPs in the configuration, AMPAvgDiskIO is computed from DiskReads and DiskWrites data as:                                                                                                                             |

| Field/<br>Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |           | $(\text{DiskReads}_1 + \text{DiskWrites}_1 + \dots + \text{DiskReads}_n + \text{DiskWrites}_n) / n$<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                   |
| HiCPUAMPUse           | FLOAT     | Highest CPUUse percentage currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                                   |
| HiCPUAMPNo            | SMALLINT  | VProcNo of an AMP with CPUUse equal to the value reported as HiCPUAMPUse.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                             |
| HiCPUAMPProc          | INTEGER   | ID of the node currently responsible for managing the AMP reported as HiCPUAMPNo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> . |
| LoCPUAMPUse           | FLOAT     | Lowest CPUUse percentage currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                                    |
| LoCPUAMPNo            | SMALLINT  | VProcNo of an AMP with CPUUse equal to the value reported as LoCPUAMPUse.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                             |
| LoCPUAMPProc          | INTEGER   | ID of the node currently responsible for managing the AMP reported as LoCPUAMPNo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> . |
| HiDiskAMP             | FLOAT     | Highest DiskUse percentage currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                                  |
| HiDiskAMPNo           | SMALLINT  | Number of an AMP with DiskUse equal to the value reported as HiDiskAMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                               |

| Field/<br>Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                |
|-----------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HiDiskAMPProc         | INTEGER   | ID of the node currently responsible for managing the AMP reported in HiDiskAMPNo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL when HiDiskAMPNo is NULL.     |
| LoDiskAMP             | FLOAT     | Lowest DiskUse percentage currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                         |
| LoDiskAMPNo           | SMALLINT  | Number of an AMP with DiskUse equal to the value reported as LoDiskAMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                     |
| LoDiskAMPProc         | INTEGER   | ID of the node currently responsible for managing the AMP reported as LoDiskAMPNo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL when LoDiskAMPNo is NULL.     |
| HiDiskAMPIO           | FLOAT     | Highest DiskReads and DiskWrites value currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                            |
| HiDiskAMPIONo         | SMALLINT  | Number of an AMP with the highest DiskReads and DiskWrites equal to the value reported as HiDiskAMPIO.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                      |
| HiDiskAMPIOProc       | INTEGER   | ID of the node currently responsible for managing the AMP reported in HiDiskAMPIONo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL when HiDiskAMPIONo is NULL. |
| LoDiskAMPIO           | FLOAT     | Lowest DiskReads and DiskWrites number currently associated with any online AMP.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                            |
| LoDiskAMPIONo         | SMALLINT  | ID of an AMP with lowest DiskReads and DiskWrites equal to the value reported as LoDiskAMPIO.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                               |

| Field/<br>Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LoDiskAMPIOProc       | INTEGER   | ID of the node currently responsible for managing the AMP reported as LoDiskAMPIONo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL when LoDiskAMPIONo is NULL.                                                         |
| PEAvgCPU              | FLOAT     | Average CPUUse for all online PEs in the configuration.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                                             |
| HiCPUPEUse            | FLOAT     | Highest CPUUse percentage currently associated with any online PE.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                                  |
| HiCPUPENo             | SMALLINT  | VProcNo of a PE with CPUUse equal to the value reported as HiCPUPEUse.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                                                                                              |
| HiCPUPEProc           | INTEGER   | ID of the node currently responsible for managing the PE reported in HiCPUPENo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL when HiCPUPENo is NULL.                                                                  |
| LoCPUPEUse            | FLOAT     | Lowest CPUUse percentage currently associated with any online PE.<br>This value is NULL if certain conditions apply. *                                                                                                                                                                                                                                                             |
| LoCPUPENo             | SMALLINT  | VProcNo of a PE with CPUUse equal to the value reported as LoCPUPEUse.<br>This value is NULL when LoCPUPEUse is NULL.                                                                                                                                                                                                                                                              |
| LoCPUPEProc           | INTEGER   | ID of the node currently responsible for managing the PE reported as LoCPUPENo.<br>This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> . |
| SessLogCount          | FLOAT     | Total number of sessions currently logged onto the system. This value is usually equal to the sum of the SessLogCount values for all PEs.<br>This value is NULL if certain conditions apply, see <a href="#">Usage Notes - MONITOR VIRTUAL SUMMARY</a> .                                                                                                                           |



| Field/<br>Column Name | Data Type                                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------|----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SesMonitorSys         | SMALLINT,<br>NOT NULL,<br>range 0-3600<br>seconds, | Maximum acceptable age of collected session-level data in memory to the PM/API application or end user.<br>This global rate is the default collection rate for all MONITOR sessions. If the value is set to zero, the collection capability is disabled.<br>This rate can be changed with the SET SESSION RATE request by specifying a system-wide option and can be saved on disk (in the version record) when it is changed.        |
| SesMonitorLoc         | SMALLINT,<br>NOT NULL,<br>range 0-3600<br>seconds, | Sets the maximum acceptable age of collected session-level data in memory for an individual Monitor partition session that submits a MONITOR SESSION request.<br>A rate of zero allows SesMonitorSys to override the current local rate for that session.<br>This rate can be changed with the SET SESSION RATE request by specifying the a LOCAL rate change option and can be saved on disk and may be lost during a system outage. |
| ResLogging            | SMALLINT,<br>NOT NULL<br>range 0-<br>3600 seconds  | Interval in seconds at which resource usage data is written to one or more active resource usage database tables.                                                                                                                                                                                                                                                                                                                     |
| ResMonitor            | SMALLINT<br>NOT NULL                               | Interval in seconds at which all resource usage data is collected in memory for reporting via the PM/API.                                                                                                                                                                                                                                                                                                                             |
| Release               | VARCHAR<br>(29)<br>NOT NULL                        | Release number of the currently running database software (for example, 15.00.00.00).<br>This value is supplied by the database.                                                                                                                                                                                                                                                                                                      |
| Version               | VARCHAR<br>(32)<br>NOT NULL                        | Version number of the currently running database software (for example, 15.00.00.00).<br>This value is supplied by the database.                                                                                                                                                                                                                                                                                                      |
| CollectionDate        | DATE<br>NOT NULL                                   | Date the Virtual Resource cache was last refreshed.                                                                                                                                                                                                                                                                                                                                                                                   |
| CollectionTime        | FLOAT<br>NOT NULL                                  | Time the Virtual Resource cache was last refreshed.                                                                                                                                                                                                                                                                                                                                                                                   |

### Sample Input - CLiv2 Request

This example shows how the parcels for a MONITOR VIRTUAL SUMMARY request, built by CLiv2, look when sent to the database. The size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body  |       |
|--------|------|--------|-------|-------|
| Num    | Name | Length | Field | Value |

| Flavor |      | Length | Body       |                         |
|--------|------|--------|------------|-------------------------|
| 0001   | Req  | 27     | Request    | MONITOR VIRTUAL SUMMARY |
| 0003   | Data | 6      | MonVerID   | 2                       |
| 0004   | Resp | 6      | BufferSize | 64000                   |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This example shows the values returned in character text format for the MONITOR VIRTUAL SUMMARY request. Your application program may display returned values in a different format.

Submitting request MONITOR VIRTUAL SUMMARY; ...

AMPAvgCPU: 24.59    AMPAvgDisk: 6.79    AMPAvgDiskIO: 1184.75

HiCPUAMPUse: 24.65    HiDiskAMP: 7.20    HiDiskAMPIO: 1218.00

HiCPUAMPNo: 0    HiDiskAMPNo: 2    HiDiskAMPIONo: 0

HiCPUAMPProc: 10001    HiDiskAMPProc: 10001    HiDiskAMPIOProc: 10001

LoCPUAMPUse: 24.53    LoDiskAMP: 6.50    LoDiskAMPIO: 1142.00

LoCPUAMPNo: 2    LoDiskAMPNo: 0    LoDiskAMPIONo: 3

LoCPUAMPProc: 10001    LoDiskAMPProc: 10001    LoDiskAMPIOProc: 10001

PEAvgCPU: 0.00

HiCPUPEUse: 0.00    LoCPUPEUse: 0.00

HiCPUPENo: 30719    LoCPUPENo: 30719

HiCPUPEProc: 10001    LoCPUPEProc: 10001

SessionCnt: 1.00

SesMonitorSys: 1    SesMonitorLoc: 0

VprocLogging: 60    VprocMonitor: 60

Release: 16u.00.00.41    Version: 16u.00.00.41\_dr182707m

Collection Date/Time: 07/20/2016 12:39:00.00

## Warning and Error Messages

All users who are logged on and issue a MONITOR VIRTUAL SUMMARY request after a system restart or after the last rate change can expect to receive one of the warnings. Typically, the types of situations that can produce warning messages are:

- After a system restart, before and after a collection period has expired.  
If the collection period has not expired and the user issues the next MONITOR VIRTUAL SUMMARY request, many of the values returned are NULL.
- After the last rate change, before and after a collection period has expired.  
If the collection period has not expired and the user issues the next MONITOR VIRTUAL SUMMARY request, many of the values returned are NULL.
- If the resource monitoring rate (ResMonitor) is not enabled, that is, if rate is set to zero. When the user issues the next MONITOR VIRTUAL SUMMARY request, many of the values returned are NULL.

## Relationship Between MONITOR VIRTUAL SUMMARY and SET RESOURCE RATE

The SET RESOURCE RATE request sets the ResMonitor and ResLogging rates, which are among the responses returned by the MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY request. Any change to either the ResMonitor or ResLogging rate results in changes in the corresponding response returned by the MONITOR VIRTUAL SUMMARY or MONITOR PHYSICAL SUMMARY request.

You must set ResMonitor to a nonzero rate for MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY to return meaningful resource utilization data. A zero ResMonitor rate returns NULL for resource utilization information.

## Relationship Between MONITOR VIRTUAL SUMMARY and SET SESSION RATE

Changes to the session-level rates (global and local) specified by SET SESSION RATE are reported in the data returned by MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY.

---

### Note:

The local rate reported is your own local rate. If the local rate is not set, the local rate is reported as zero.

---

As more session-level monitoring is done (by setting a faster SET SESSION RATE), the resulting overhead may increase the level of CPU usage (reported in MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY data) by your system. However, this may depend on the size of the rate change and the type of work done by other sessions.

## Relationship Between MONITOR VIRTUAL SUMMARY and MONITOR VIRTUAL CONFIG

Use MONITOR VIRTUAL SUMMARY with the MONITOR VIRTUAL CONFIG request for an overall system status. These are low overhead requests.

- Execute the MONITOR VIRTUAL SUMMARY request every 5 or 10 minutes for a low-cost, continuous monitoring of your system.
- Execute the MONITOR VIRTUAL CONFIG request to get a picture of your system configuration at defined times, such as at the beginning of a day, various times during the day, or when the system is down.

For information on this PMPC PM/API request relationship, see [Relationship Between MONITOR VIRTUAL CONFIG and MONITOR VIRTUAL SUMMARY](#).

## MONITOR WD

Returns a subset of the RSS ResUsageSps data.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data. If data is supplied for that field, set the bit to zero. If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This must be version 9 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                                                                                                                                                                                 |

### Monitor Privileges

To use this request, you must have the MONRESOURCE privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

### Usage Notes - MONITOR WD

The data returned through MONITOR WD is ResUsageSps data from the RSS memory buffer.

You can use Teradata Viewpoint, CNS commands via Database Window, or SET RESOURCE RATE to enable RSS collection.

The data in the buffer is summarized to unique service-level-goal-driven Priority Scheduler workload definition ID (pWDId) and VprType field values. For a description of the pWDId field, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099. For details on the service-level-goal-driven Priority Scheduler WD, see *Teradata® Viewpoint User Guide*, B035-2206.

The WDId and pWDId fields return valid ID values.

---

**Note:**

Zero is a valid value for the pWDId field.

---

TASM Workloads rule is always enabled.

For information on TASM rules, see *Teradata® Viewpoint User Guide*, B035-2206.

## CLiv2 Response Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                   | Comments/Key Parcel Body Fields                                                                                                                              |
|-----------------|---------------|------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                        | Statement No =1<br>ActivityCount = 1<br>ActivityType = PCLMONWDRESRSTMT (202)                                                                                |
| DataInfo        | 71            | 6 to 64100                                                                                                       | Optional: this parcel is present if request was IndicData parcel.                                                                                            |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>11 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. See <a href="#">Statement 1</a> for details on the data returned. |
| EndStatement    | 11            | 6                                                                                                                | StatementNo = 2-byte integer                                                                                                                                 |
| Success         | 8             | 18 to 273                                                                                                        | Statement No =2<br>ActivityCount = Number of Record parcels in statement 2<br>ActivityType = PCLMONWDRESRSTMT (202)                                          |
| DataInfo        | 71            | 6 to 64100                                                                                                       | Optional: this parcel is present if request was IndicData parcel.                                                                                            |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>11 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in Record or indicator mode. See <a href="#">Statement 2</a> for details on the data returned. |
| EndStatement    | 11            | 6                                                                                                                | StatementNo = 2-byte integer                                                                                                                                 |
| End Request     | 12            | 4                                                                                                                | None                                                                                                                                                         |

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Statement 1

The response to the first statement results in a Record parcel containing the fields below.

| Field/Column Name | Data Type            | Description                                    |
|-------------------|----------------------|------------------------------------------------|
| SampleSec         | SMALLINT<br>NOT NULL | Duration of the collection period in seconds.  |
| AMPNodes          | SMALLINT<br>NOT NULL | Number of nodes with at least one online AMP.  |
| PENodes           | SMALLINT<br>NOT NULL | Number of nodes with at least one online PE.   |
| CollectionDate    | DATE,<br>NOT NULL    | Date the WD resource cache was last refreshed. |
| CollectionTime    | FLOAT<br>NOT NULL    | Time the WD resource cache was last refreshed. |

### Statement 2

The second statement of the response results in multiple Record parcels.

For more information on the following columns, see the ResUsageSps Table and ResSpsView in *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

| Field/Column Name | Data Type               | Description                                                                                          |
|-------------------|-------------------------|------------------------------------------------------------------------------------------------------|
| PPId              | SMALLINT<br>NOT NULL    | This field is obsolete and returns a value of zero.                                                  |
| PGId              | SMALLINT<br>NOT NULL    | This field returns the pWDId value.                                                                  |
| VprType           | VARCHAR (4)<br>NOT NULL | Type of vproc: <ul style="list-style-type: none"> <li>• AMP</li> <li>• PE</li> <li>• MISC</li> </ul> |

| Field/Column Name | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WDId              | INTEGER<br>NOT NULL  | WD ID. On SLES 11 or later systems, TASM Workloads rule is always enabled. For information on TASM rules, see <i>Teradata® Viewpoint User Guide</i> , B035-2206.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| AGId              | SMALLINT<br>NOT NULL | This field is obsolete and returns a value of zero.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| RelWgt            | SMALLINT<br>NOT NULL | This field is obsolete and returns a value of zero.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| NumTasks          | INTEGER<br>NOT NULL  | Average number of tasks of online nodes. The field is the result of:<br>$\text{NumTasks} = \text{SUM of } (\text{NumTasks-}i) / N$ where:<br><i>NumTasks-i</i> is the number of tasks assigned to the WD at the end of the reporting period.<br><i>i</i> varies from 1 to <i>N</i> , where <i>N</i> is the number of online nodes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| QWaitTime         | FLOAT<br>NOT NULL    | Total wait time in milliseconds that work requests waited on an input queue before being serviced.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| QWaitTimeMax      | FLOAT<br>NOT NULL    | Maximum time in milliseconds that work requests waited on an input queue before being serviced.<br>The field is the result of:<br>$\text{QWaitTimeMax} = \text{MAX } (\text{QWaitTimeMax-}i)$ where: <ul style="list-style-type: none"> <li><i>QWaitTimeMax-i</i> is <i>QWaitTimeMax</i> in each online node.</li> <li><i>i</i> varies from 1 to <i>N</i>, where <i>N</i> is the number of online nodes.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| CPUUserPct        | FLOAT<br>NOT NULL    | Weighted average of CPUUserPct of each node.<br>This field is the result of:<br>$\text{CPUUserPct} = \text{SUM of } (\text{CPUUserPct-}i * \text{ScalingFactor-}i) / \text{SUM of } (\text{ScalingFactor-}i)$ where: <ul style="list-style-type: none"> <li><i>CPUUserPct-i</i> is calculated as:<br/> <math display="block">(\text{CPUUServAwt} + \text{CPUUServDisp} + \text{CPUUServMisc} + \text{CPUUExecAwt} + \text{CPUUExecDisp} + \text{CPUUExecMisc}) * 100 / (NCPUs * \text{Centisecs} * 10)</math> <i>NCPUs</i> is the number of CPUs in the node.</li> <li><i>i</i> varies from 1 to <i>N</i>, where <i>N</i> is the number of online nodes.</li> <li><i>ScalingFactor-i</i> is the node CPU normalization factor in each node.</li> </ul> <p><b>Note:</b><br/>           The CPU times are in milliseconds.<br/>           The Parser CPU times are included in the Dispatcher CPU times.</p> |

| Field/Column Name | Data Type           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WorkMsgMaxDelay   | FLOAT<br>NOT NULL   | General indicator only. This field is result of the following calculation:<br>$WorkMsgMaxDelay = \text{MAX} (WorkMsgMaxDelay-i)$<br>where:<br><ul style="list-style-type: none"> <li><math>WorkMsgMaxDelay-i</math> is calculated in each online node as:<br/> <math>WorkMsgsendDelayMax + WorkMsgReceiveDelayMax</math></li> <li><math>i</math> varies from 1 to <math>N</math>, where <math>N</math> is the number of online nodes.</li> </ul> <b>Note:</b><br>WorkMsgMaxDelay does not represent the subtotal of the same message on the send and receive side.           |
| WorkTypeInuseMax  | INTEGER<br>NOT NULL | Total of the AMP Worker Task (AWT) columns:<br>$WorkTypeInuseMax = \text{MAX} (WorkTypeInuseMax-i)$<br>where:<br><ul style="list-style-type: none"> <li><math>WorkTypeInuseMax-i</math> is the sum of WorkTypeMax00 through WorkTypeMax15 in each node.</li> <li><math>i</math> varies from 1 to <math>N</math>, where <math>N</math> is the number of online nodes.</li> </ul>                                                                                                                                                                                              |
| WorkTimeInuseAvg  | FLOAT<br>NOT NULL   | Average number of AWTs used. This field is result of:<br>$WorkTimeInuseAvg = \text{SUM of } (WorkTimeInuse-i) / N$<br>where:<br><ul style="list-style-type: none"> <li><math>WorkTimeInuse-i</math> is calculated in each online node as:<br/> <math>WorkTimeInuse / (\text{Centisecs} * 10 * NCPUs)</math><br/> <math>NCPUs</math> is the number of CPUs in the node.</li> <li><math>i</math> varies from 1 to <math>N</math>, where <math>N</math> is the number of online nodes.</li> </ul> <b>Note:</b><br>This value is available in the ResSpsView view as AwtUsedAvg. |
| IODelay           | FLOAT<br>NOT NULL   | Number of I/Os that are delayed. This field is result of:<br>$\text{ProcBlksFsgRead} + \text{ProcBlksFsgWrite} + \text{ProcBlksFsgNIOs}$                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| IODelayTime       | FLOAT<br>NOT NULL   | Total time the I/O is delayed for. This field is the result of:<br>$\text{ProcWaitFsgRead} + \text{ProcWaitFsgWrite} + \text{ProcWaitFsgNIOs}$                                                                                                                                                                                                                                                                                                                                                                                                                               |
| PhysicalRead      | FLOAT<br>NOT NULL   | Number of physical reads performed for this period. This field is the result of:<br>$\text{FilePDbAcqReads} + \text{FilePDbPreReads} + \text{FilePCiAcqReads} + \text{FileSdbAcqReads} + \text{FileSCiAcqReads} + \text{FileTJtAcqReads} + \text{FileAPtAcqReads} + \text{FilePCiPreReads} + \text{FileSdbPreReads} + \text{FileSCiPreReads} + \text{FileTJtPreReads} + \text{FileAPtPreReads}$                                                                                                                                                                              |
| PhysicalReadKB    | FLOAT<br>NOT NULL   | Number of physical reads in KB performed for this period. This field is result of:<br>$\text{FilePDbAcqReadKB} + \text{FilePDbPreReadKB} + \text{FilePCiAcqReadKB} + \text{FileSdbAcqReadKB} + \text{FileSCiAcqReadKB} + \text{FileTJtAcqReadKB} + \text{FileAPtAcqReadKB} + \text{FilePCiPreReadKB} + \text{FileSdbPreReadKB} + \text{FileSCiPreReadKB} + \text{FileTJtPreReadKB} + \text{FileAPtPreReadKB}$                                                                                                                                                                |



| Field/Column Name | Data Type         | Description                                                                                                                                                                                                                                                                  |
|-------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |                   | FileSdbPreReadKB + FileSCiPreReadKB + FileTJtPreReadKB + FileAPtPreReadKB                                                                                                                                                                                                    |
| PhysicalWrite     | FLOAT<br>NOT NULL | Number of physical writes performed for this period. This field is result of:<br>FilePDbFWrites + FilePCiFWrites + FileSdbFWrites + FileSCiFWrites + FileTJtFWrites + FileAPtFWrites                                                                                         |
| PhysicalWriteKB   | FLOAT<br>NOT NULL | Number of physical writers in KB performed for this period. This field is result of:<br>FilePDbFWriteKB + FilePCiFWriteKB + FileSdbFWriteKB + FileSCiFWriteKB + FileTJtFWriteKB + FileAPtFWriteKB                                                                            |
| LogicalRead       | FLOAT<br>NOT NULL | Number of logical reads performed for this period. This field is result of:<br>FilePDbAcqs + FilePDbPres + FilePCiAcqs + FileSdbAcqs + FileSCiAcqs + FileTJtAcqs + FileAPtAcqs + FilePCiPres + FileSdbPres + FileSCiPres + FileTJtPres + FileAPtPres                         |
| LogicalReadKB     | FLOAT<br>NOT NULL | Number of logical reads in KB performed for this period. This field is result of:<br>FilePDbAcqKB + FilePDbPresKB + FilePCiAcqKB + FileSdbAcqKB + FileSCiAcqKB + FileTJtAcqKB + FileAPtAcqKB + FilePCiPresKB + FileSdbPresKB + FileSCiPresKB + FileTJtPresKB + FileAPtPresKB |
| LogicalWrite      | FLOAT<br>NOT NULL | Number of logical writes performed for this period. This field is result of:<br>FilePDbDyRRels + FilePCiDyRRels + FileSdbDyRRels + FileSCiDyRRels + FileTJtDyRRels + FileAPtDyRRels                                                                                          |
| LogicalWriteKB    | FLOAT<br>NOT NULL | Number of logical writes in KB performed for this period. This field is result of:<br>FilePDbDyRRelKB + FilePCiDyRRelKB + FileSdbDyRRelKB + FileSCiDyRRelKB + FileTJtDyRRelKB + FileAPtDyRRelKB                                                                              |
| VPId              | FLOAT<br>NOT NULL | Virtual partition ID.                                                                                                                                                                                                                                                        |
| WaitIO            | FLOAT<br>NOT NULL | Number of milliseconds tasks in WD waited for I/O over the reporting period.<br>WaitIO is updated when the wait for I/O is completed.                                                                                                                                        |
| WaitOther         | FLOAT<br>NOT NULL | Number of milliseconds tasks in WD waited for reasons other than I/O over the reporting period (for example, a task waiting for a message).<br>WaitOther is updated when wait is completed.                                                                                  |
| CPURunDelay       | FLOAT<br>NOT NULL | Number of milliseconds tasks in the WD sat in the CPU runqueue waiting to run over the reporting period.<br>This data can be used in determining demand for the virtual partition and Workload Share Percent. The Workload Share                                             |

| Field/Column Name     | Data Type         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |                   | <p>Percent is a workload management method.*If the CPU and I/O percentages for a virtual partition or WD are below their relative share values and the CPURunDelay values are low, there was insufficient demand to meet the share percentage. If the CPURunDelay values are high, higher tier SQL requests were allocated more resources so that there were insufficient resources remaining to allocate to SQL requests in this WD to meet its relative share.</p> <p><b>Note:</b><br/>A virtual partition divides a system so that a percentage of resources are allocated to a collection of workloads. A virtual partition can consist of WDs from all management methods.</p> |
| IOSubmitted           | FLOAT<br>NOT NULL | Number of I/Os submitted on behalf of this WD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| IOSubmittedKB         | FLOAT<br>NOT NULL | KB of I/O submitted on behalf of this WD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| IOCompleted           | FLOAT<br>NOT NULL | Number of AgeOut Now data blocks not to keep in memory (fsgcache) and to be written to disk.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| IOCompletedKB         | FLOAT<br>NOT NULL | KB of AgeOut Now data blocks not to keep in memory (fsgcache) and to be written to disk.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| IOCriticalSubmitted   | FLOAT<br>NOT NULL | Number of I/Os submitted with critical status. These I/Os execute at top priority instead of being based on the I/O priority of the SQL request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| IOCriticalSubmittedKB | FLOAT<br>NOT NULL | KB of I/O submitted with critical status. These I/Os execute at top priority instead of being based on the I/O priority of the SQL request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| DecayLevel1IO         | FLOAT<br>NOT NULL | <p>Number of times SQL requests in the WD hit decay level 1 due to I/O.</p> <p><b>Note:</b><br/>DecayLevel1IO is used for Timeshare WDs** only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| DecayLevel2IO         | FLOAT<br>NOT NULL | <p>Number of times SQL requests in the WD decay level 2 due to I/O.</p> <p><b>Note:</b><br/>DecayLevel2IO is used for Timeshare WDs** only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| DecayLevel1CPU        | FLOAT<br>NOT NULL | <p>Number of times SQL requests in the WD hit decay level 1 due to CPU.</p> <p><b>Note:</b><br/>DecayLevel1CPU is used for Timeshare WDs** only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Field/Column Name    | Data Type         | Description                                                                                                                                                           |
|----------------------|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DecayLevel2CPU       | FLOAT<br>NOT NULL | Number of times SQL requests in the WD hit decay level 2 due to CPU.<br><br><b>Note:</b><br>DecayLevel2CPU is used for Timeshare WDs** only.                          |
| TacticalExceptionIO  | FLOAT<br>NOT NULL | Number of times SQL requests in the WD hit a tactical per-node exception due to I/O.<br>An exception, used only for Tactical WDs, is created for each Tactical WD***. |
| TacticalExceptionCPU | FLOAT<br>NOT NULL | Number of times SQL requests in the WD hit a tactical per-node exception due to CPU.<br><br><b>Note:</b><br>TacticalExceptionCPU is used for Tactical WDs***.         |

\* The Workload Share Percent Management Method workload is assigned a proportion of the resources that are available after allocations have been made for tactical workloads. The percentage of resources is divided equally between all requests running in the WD. For example, if the Workload Share Percent is 5% and there are five SQL requests, each SQL request will get 1% of the share resources. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

\*\* The Timeshare Workload Management Method workload can be assigned to one of four stepped access levels, Top, High, Medium, or Low. The higher access levels are given larger access rates than the lower levels. For example, an SQL request assigned to a Timeshare WD with a Top access level, which has an access rate of 8, would receive eight times the amount of resources than an SQL request assigned to a Low access level.

Timeshare workloads are assigned resources remaining after all allocations have been made for tactical and Workload Share Percent workloads. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

\*\*\* The Tactical Workload Management Method workload yields the fastest available response time and executes at the highest tier, preempting all resource needs of other tiers. This method is well suited for critical, short-running queries that require fast response times. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

### Sample Input - CLlv2 Request

The following example shows how the parcels for a MONITOR WD request, built by CLlv2, appear when sent to the database server.

---

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

---

| Flavor |      | Length | Body        |            |
|--------|------|--------|-------------|------------|
| Num    | Name | Bytes  | Field       | Value      |
| 0001   | Req  | 14     | Request     | MONITOR WD |
| 0003   | Data | 6      | MonVerID    | 9          |
| 0004   | Resp | 6      | Buffer Size | 64000      |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

The MONITOR WD request returns values approximately as shown below when TASM Workloads are enabled (see *Teradata® Viewpoint User Guide*, B035-2206 for information on this rule):

#### Note:

The Monitor WD request commonly returns values in text character format. Your application program may return the values in a different format or display.

```

ResRate: 30;      AMP Nodes: 1;   PE Nodes: 1
Collection Date/Time: 06/15/2011 18:34:01.00
PGId VT  PP  CPUUsrPct QWaitTime QWTimeMax WkMsgMaxD WTypeMax  WTimeAvg
WDId AGId RWgt IODelay  IODelayTi PhyRead  PhyReadMB PhyWrite  PhyWriteMB
      NPrc Reserved1 Reserved2 LogRead  LogReadMB LogWrite  LogWriteMB
      VPID WaitIO    WaitOther CPURunDly
      IOsubmit IOSubmKB  IOComple IOComplKB IOCriticl IOCritKB
      Decay1IO Decay2IO  Decay1CPU Decay2CPU TacExcpIO TacExcpCPU
=====
SUCCESS parcel:
StatementNo=2, ActivityCount=10,
ActivityType=202, FieldCount=39
  0 AMP      0      98.56      0.00      0.00      0.00      4      3.99
 12  0      0      0.00      0.00      0.00      0.00      0.00      0.00
    132      0.00      0.00      0.00      0.00      0.00      0.00
      1    300.00    7558.00  82602.00
        151.00    9664.00    151.00    9664.00      0.00      0.00
          0.00      0.00      0.00      0.00      0.00      0.00
-----
  0 PE      0      0.00      0.00      0.00      0.00      0      0.00

```

|       |      |       |        |            |         |        |      |      |
|-------|------|-------|--------|------------|---------|--------|------|------|
| 12    | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 33    | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 1     | 0.00   | 0.00       | 0.00    |        |      |      |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 250   | AMP  | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 1    | 1.00 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 2288  | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 100   | 0.00   | 119692.00  | 0.00    |        |      |      |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 251   | MISC | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0    | 0.00 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 160   | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 100   | 0.00   | 0.00       | 0.00    |        |      |      |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 251   | AMP  | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0    | 0.00 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 892   | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 100   | 0.00   | 0.00       | 0.00    |        |      |      |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 254   | MISC | 0     | 0.32   | 0.00       | 0.00    | 0.00   | 0    | 0.00 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 4400  | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 102   | 0.00   | 218950.00  | 318.00  |        |      |      |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 254   | AMP  | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0    | 0.01 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 79844 | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 102   | 359.00 | 3511485.00 | 1701.00 |        |      |      |
|       |      |       | 38.00  | 228.00     | 38.00   | 228.00 | 0.00 | 0.00 |
|       |      |       | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
| ----- |      |       |        |            |         |        |      |      |
| 254   | PE   | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0    | 0.00 |
| 0     | 0    | 0     | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |
|       |      | 58034 | 0.00   | 0.00       | 0.00    | 0.00   | 0.00 | 0.00 |

|       |      |      |  |          |            |          |            |      |
|-------|------|------|--|----------|------------|----------|------------|------|
|       |      | 102  |  | 272.00   | 2644347.0  | 399.00   |            |      |
|       |      |      |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      |      |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
| ----- |      |      |  |          |            |          |            |      |
| 255   | MISC | 0    |  | 0.09     | 0.00       | 0.00     | 0.00       | 0    |
| 0     | 0    | 0    |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      | 2992 |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      | 101  |  | 0.00     | 548164.00  | 2922.00  |            |      |
|       |      |      |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      |      |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
| ----- |      |      |  |          |            |          |            |      |
| 255   | AMP  | 0    |  | 0.03     | 0.00       | 0.00     | 0.00       | 0    |
| 0     | 0    | 0    |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      | 3520 |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      | 101  |  | 0.00     | 315555.00  | 45137.00 |            |      |
|       |      |      |  | 13567.00 | 1722001.00 | 13569.00 | 1722255.00 | 0.00 |
|       |      |      |  | 0.00     | 0.00       | 0.00     | 0.00       | 0.00 |
|       |      |      |  |          |            |          |            |      |
|       |      |      |  |          |            |          |            |      |
|       |      |      |  |          |            |          |            |      |

## SET RESOURCE RATE

Sets either the ResMonitor or ResLogging rate.

### Input Data

| Element            | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>     | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode. |
| <i>mon_ver_id</i>  | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 2 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |
| <i>sample_rate</i> | SMALLINT<br>NOT NULL | Value of the collection rate. This field is used either to collect resource data or to log resource data to the resource usage tables.<br>You can specify one of the following:                                                                                                                                                                                                                                                                               |

| Element               | Data Type         | Description                                                                                                                                                                                                                                                                                                                            |
|-----------------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | range 0-3600 secs | <ul style="list-style-type: none"> <li>The ResMonitor rate value if you want to change the resource monitoring rate.</li> <li>The ResLogging rate value if you want to change the resource logging rate.<br/>The value should be an integral divisor of 3600.</li> <li>Zero to turn off the resource collection or logging.</li> </ul> |
| <i>log_change</i>     | VARCHAR (1)       | Indicator of whether this rate applies to the ResLogging or ResMonitor rate: <ul style="list-style-type: none"> <li>Y or y = ResLogging rate</li> <li>N, n, NULL, or blank = ResMonitor rate</li> </ul>                                                                                                                                |
| <i>virtual_change</i> | VARCHAR (1)       | <b>Note:</b><br>This field is obsolete.                                                                                                                                                                                                                                                                                                |

## Monitor Privileges

To use this request, you must have the SETRESRATE privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- Teradata Vantage™ - Database Administration, B035-1093
- Teradata Vantage™ - Advanced SQL Engine Security Administration, B035-1100
- Teradata JDBC Driver Reference, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - SET RESOURCE RATE

You can set the ResMonitor or ResLogging rates using the SET RESOURCE RATE request. For a description of these rates, see [Data Collection](#).

ResMonitor and ResLogging are independent of each other because you can monitor without wanting to log to the resource usage tables, or you can log without monitoring.

### Note:

You can set only one ResMonitor or ResLogging rate within one SET RESOURCE RATE request. For example, to set both the ResMonitor and ResLogging rates at the same time, you must issue two SET RESOURCE RATE requests and specify the USING Data String appropriately.

Resource data is placed in a memory repository separate and independent from session usage data. Therefore, any changes in the ResMonitor rate does not impact session usage data.

Remember that resource utilization data collected by the ResMonitor rate is collected and reported differently from session utilization data. Whereas session usage data is collected cumulatively, resource

data is collected for a particular collection period. The data reported is based on the activity that occurred during that collection period and does not include any cumulative data over collection periods.

There is a difference between saving statistics in the resource memory repository and returning the data for display. Resource data can be saved after a collection rate is set to a nonzero rate, but no return of data occurs until a MONITOR request is issued.

The ResMonitor rate and the ResLogging rate are saved on disk in the Version Record when they are changed. For this reason, the SET RESOURCE RATE request can block if someone else is updating the GDO control record. If a block occurs, you must wait until the block clears.

---

**Note:**

When you execute the SET RESOURCE RATE request, the change is saved in the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view) and written to the system console running Database Windows.

---

## CLiv2 Response Parcels

The response returned from the database contains the following sequence of parcel types.

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                                                  |
|-----------------|---------------|----------------|----------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273      | Activity Count = Contains previous rate.<br>Activity Type = 87 (PCLSETRESSR)     |
| DataInfo        | 71            | 6 to 64100     | This parcel is present if request was IndicReq parcel; depends on the data type. |
| EndStatement    | 11            | 6              | StatementNo: 2-byte integer                                                      |
| EndRequest      | 12            | 4              | None                                                                             |

## Sample Input - CLiv2 Request

This example shows how the parcels for a SET RESOURCE RATE request, built by CLiv2, look when sent to the database server using a *sample\_rate* of 600 seconds and a *logging\_change* of Y. The size of the response buffer is set in the example at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body                                 |                   |
|--------|------|--------|--------------------------------------|-------------------|
| Num    | Name | Bytes  | Field                                | Value             |
| 0001   | Req  | 21     | Request                              | SET RESOURCE RATE |
| 0030   | Data | 11     | MonVerID<br>SampleRate<br>LoggingChg | 2<br>600<br>Y     |



| Flavor |      | Length | Body        |       |
|--------|------|--------|-------------|-------|
|        |      |        | VirtualChg  | N     |
| 0004   | Resp | 6      | Buffer Size | 64000 |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

With a *sample\_rate* of 600 and a *logging\_change* of y, this example shows the values returned in character text format for the SET RESOURCE RATE request. Your application program may display returned values in a different format.

```
Success parcel:
  StatementNo: 1      ActivityCount: 60
  ActivityType: 87    FieldCount: 0
DataInfo parcel:
  FieldCount: 0
EndStatement.
EndRequest.
```

### Relationship Between SET RESOURCE RATE and MONITOR PHYSICAL RESOURCE or MONITOR VIRTUAL RESOURCE

You must execute the SET RESOURCE RATE request to activate resource data collection before you execute a MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request. This means that you must set the ResMonitor rate to nonzero. If the ResMonitor rate is set to zero, you will receive an error message.

A change in the resource collection rate by User A, for example, may affect the data reported by MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request made by User B. If the ResMonitor rate is altered, User B receives a warning message when executing a subsequent MONITOR VIRTUAL RESOURCE or MONITOR PHYSICAL RESOURCE request.

### Relationship Between SET RESOURCE RATE and MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY

The SET RESOURCE RATE request sets the ResMonitor and ResLogging rates, which are among the responses returned by the MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY request. Any change to either the ResMonitor or ResLogging rate results in changes in the corresponding response returned by the MONITOR VIRTUAL SUMMARY or MONITOR PHYSICAL SUMMARY request.

You must set ResMonitor to a nonzero rate for MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY to return meaningful resource utilization data. A zero ResMonitor rate returns NULL for resource utilization information.

## SET SESSION ACCOUNT

Changes the account string for the session or for the request.

### Input Data

| Element           | Data Type                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                         | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.                                                                                                                                               |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL         | MONITOR software version ID. This can be version 2 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <i>host_id</i>    | SMALLINT<br>NOT NULL         | ID of the host upon which the session was issued. <i>host_id</i> cannot exceed 1023. A <i>host_id</i> of zero identifies the database operator console. If you do not specify a valid <i>host_id</i> , an error message is returned to CLlv2. Use only if you have DBA privileges.                                                                                                                                                                                                                                                                                                                          |
| <i>session_no</i> | INTEGER<br>NOT NULL          | Number of the session. <i>session_no</i> combined with the <i>host_id</i> produces a unique Session ID. If you do not specify a valid <i>session_no</i> , an error message is returned to CLlv2. Use only if you have DBA privileges.                                                                                                                                                                                                                                                                                                                                                                       |
| <i>account</i>    | VARCHAR<br>(512)<br>NOT NULL | Account string for the session or request.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <i>sess_req</i>   | VARCHAR (1)                  | Indicator of how the new account or priority affects requests for a specified session.<br>If you specify Y or y, the change applies to all current and future requests for a specified session. If no requests or steps are executing, the new account/priority takes effect at the next request, and the DBC.SessionTbl table reflects the new account/priority for the current session.<br>If you specify NULL, blank, N, or n, the change applies to the current request for the specified session. If no request is executing, the next request for the specified session has the old account/priority. |

## Monitor Privileges

To use this request, you must have the ABORTSESSION or an equivalent privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - SET SESSION ACCOUNT

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

The account or priority change is recorded in the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view) with the following text in the TEXT column:

```
SESSION session_no HOSTID host_id CHANGED FROM ACCOUNT account TO ACCOUNT
account ON sess_req
```

The EVENT\_TAG field contains an event number. THEDATE and THETIME fields, which make up the index of the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view), contain the date and time of the account/priority change. All other fields of the table are blank.

When TASM Workloads are enabled, the SET SESSION ACCOUNT request will:

- Fail and return an error.
- Or succeed for a session, but the request in which it is running will continue to run the old (existing) account string. Future requests will run with the new account string.

For information on TASM rules, see *Teradata® Viewpoint User Guide*, B035-2206.

## CLIV2 Response Parcels

The response returned from the database contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes)             | Comments/Key Parcel                                                              |
|-----------------|---------------|----------------------------|----------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                  | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = 108 (SET SESSION ACCOUNT) |
| DataInfo        | 71            | 6 to 64100                 | Optional; this parcel is present if request was IndicReq parcel.                 |
| Record          | 10            | • 5 to 64100 (record mode) | This record contains the old account and an error code.                          |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                | Comments/Key Parcel          |
|-----------------|---------------|-------------------------------------------------------------------------------|------------------------------|
|                 |               | <ul style="list-style-type: none"> <li>6 to 64100 (indicator mode)</li> </ul> |                              |
| EndStatement    | 11            | 6                                                                             | StatementNo = 2-byte integer |
| EndRequest      | 12            | 4                                                                             | None                         |

The Data parcel sent from the host should be 39 bytes long for record mode or 40 bytes long for indicator mode.

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The statement response results in a Record parcel containing:

### Note:

The *Column Name* and *Column Contents* field values are not returned in the Record parcel. These values are returned in an IDENTIFY request.

| Field/Column Name | Data Type                                          | Description              |
|-------------------|----------------------------------------------------|--------------------------|
| OldAccount        | VARCHAR (128)<br>CHARACTER SET UNICODE<br>NOT NULL | Existing account string. |
| ErrorCode         | INTEGER<br>NOT NULL                                | An error code.           |

The Error Code column can contain any of the following return codes:

| Error-Name      | Code | Text                                                     |
|-----------------|------|----------------------------------------------------------|
| ErrPFMNoAr      | 3250 | No access right.                                         |
| ErrPFMBadSes    | 3256 | User entered invalid session x HostId.                   |
| ErrPFMBadAcc    | 3292 | User entered invalid account.                            |
| ErrPFMUpdSesTbl | 3293 | Failed to update the session table with the new account. |

| Error-Name         | Code | Text                                                             |
|--------------------|------|------------------------------------------------------------------|
| ErrPFMBadSesReqInd | 3294 | Invalid session/request indicator value for set session account. |
| ErrPFMInvSes       | 3295 | Invalid session.                                                 |

### Sample Input - CLlv2 Request

This is an example of the request parcels sent from a mainframe client. It shows how the parcels for a SET SESSION ACCOUNT request, built by CLlv2, look when sent to the database server, where *account* is \$Hacct, *sess\_req* is Y, *host\_id* is 348, and *session\_no* is 1000.

In this sample, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body                      |                     |
|--------|------|--------|---------------------------|---------------------|
| Num    | Name | Bytes  | Field                     | Value               |
| 0001   | Req  | 22     | Request                   | SET SESSION ACCOUNT |
| 0003   | Data | 45     | MonVerID                  | 2                   |
|        |      |        | HostId                    | 348                 |
|        |      |        | SessionNo                 | 1000                |
|        |      |        | Account                   | \$Hacct             |
|        |      |        | Session/Request Indicator | Y                   |
| 0004   | Resp | 6      | BufferSize                | 64000               |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

This sample shows typical values returned in character text format for SET SESSION ACCOUNT. Your application may return values in a different format.

```

Success Parcel:
  Statement No: 1 Activity Count: 1
  Activity Type: 108 FieldCount: 2
DataInfo Parcel:
  Field Count: 2
Record Parcel:

```

```
Parcel Flavor: 10 Parcel Body Length:32
OldAcct: "$MAcct", ErrorCode = 0
EndStatement.EndRequest.
```

## SET SESSION RATE

Sets the global and local rates for updating session-level statistics in memory.

### Input Data

| Element             | Data Type                                       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>      | BYTE                                            | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode. |
| <i>mon_ver_id</i>   | SMALLINT<br>NOT NULL                            | MONITOR software version ID. This can be version 2 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |
| <i>sample_rate</i>  | SMALLINT<br>NOT NULL<br>range 1-3600<br>seconds | Value of the collection interval.<br><br><b>Note:</b><br>If the global sample rate is set to zero an error is returned.                                                                                                                                                                                                                                                                                                                                       |
| <i>local_change</i> | VARCHAR (1)                                     | Type of session to which this rate change applies.<br>Specify Y or y if the rate is set within a local session or N, n, NULL, or blank if the rate applies to global queries.                                                                                                                                                                                                                                                                                 |

### Monitor Privileges

To use this request, you must have the SETSESSRATE privilege as part of your default role or this privilege must be granted directly to you.

If you make changes to either the system or local rate, this can reset the starting point at which data is collected. Therefore, Teradata recommends that the SETSESSRATE privilege be granted to a restricted number of users (such as DBAs or certain application programmers)

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100

- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes - SET SESSION RATE

This request is recorded in the DBC.SW\_Event\_Log table (accessible from the DBC.Software\_Event\_LogV view) and also written to the DBW. Global rate updates are recorded only in the GDO control record.

Session usage data is placed in a separate, independent memory repository from the processor resource data. As a result, any changes in the session monitoring rate have no impact on the resource data. However, a user who sets a local collection rate could affect the session usage data that other users see because all session usage data is stored in the same memory repository.

Session-level usage data is collected differently from the way processor resource usage data is collected. Processor resource usage data is collected for a particular collection period, while session usage data is collected cumulatively. The report can include cumulative data over many collection periods.

Monitoring a system may cause some form of global performance degradation. To limit the overhead cost, Teradata recommends that you set the global rate at 3600 seconds (or 1 hour) for general system monitoring. To perform problem analysis more quickly, you can set a more frequent local rate. After the analysis is done, terminate the session or set the local rate to zero. The lower global rate takes effect again in that session.

To avoid confusion, Teradata recommends that applications not query data more frequently than the faster of the two rates. If MONITOR SESSION requests are executed too frequently, one of the following may occur:

- The user application program sees duplicated data and CPU resources are wasted.
- The user application program sees data showing the result of an alteration caused by another user.

It is also recommended that you restrict the SET SESSION RATE privilege to the database administrator or system administrator.

A change to the global rate may cause a block because:

- Someone else is changing the global rate at the same time.  
Rate change requests are queued and processed in the order received because the system does not allow different rates to be used on different processors. The system ensures that all processors use the same rate.
- The GDO control record is being updated.

## Rate Scenario 1: Effects on Other Users When Increasing the Global Rate

User A has the SET SESSION RATE privilege. User B does not.

| Step | Action                                                                                                                               |
|------|--------------------------------------------------------------------------------------------------------------------------------------|
| 1    | User A changes the global rate from 10 minutes to 5 minutes at 8:55. User A now requests data every 5 minutes instead of 10 minutes. |

| Step | Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2    | User A makes a request at 8:57. Data returned shows data from the interval of 8:50 to 8:57, plus the cumulative data from 8:00 to 8:50. By forcing an update of the data at 8:57, User A has reset the collection start time (or time-stamp) to 8:57 because the current time (8:57) minus the time-stamp of the data (8:50) is greater than the session collection rate of 5 minutes. Resetting the collection start time means resetting the time from which the interval is calculated to 8:57. |
| 3    | User A makes a next request at 9:00. Data returned is the same data that was returned at 8:57. The reason is that the current session-level data that was available was considered current, that is, the age of the data (9:00 - 8:57 = 3) is less than the session collection rate of 5.                                                                                                                                                                                                          |

If User B is not aware that the global rate changed to 5 minutes, the application for User B continues to request data every 10 minutes. User B is requesting data at a less frequent rate than the data collection rate. User B may not notice a difference in data, since the data is cumulative. However, since all session resource data is deposited in the same memory pool, a change in collection rate by another user could affect the data User B sees.

Although it is transparent to the user, the user data impacted depends on which user request is received first. This usually happens when the user is not requesting data at the same rate as data is collected.

For example, User A makes a request at 9:00 and the request made by User A is received before the request (also at 9:00) made by User B. As described previously, User A sees data returned that is cumulative from 8:00 to 8:57.

| Step | Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | The request made by User B at 9:00 is queued. User B gets the cumulative data from 8:00 to 8:57 because User A made a request at 8:57. The request made by User B does not trigger an update because the request made by User A resets the time-stamp (or start time) to 8:57, and the elapsed time between 8:57 to 9:00 is less than 5 minutes (the data collection rate of the session). Thus, User B gets data from 8:50 to 8:57, plus the cumulative data from 8:00 to 8:50. In this case, there is no difference in the data User A or User B sees as a result of the request order. |
| 2    | User B gets a warning message that the rate changed to 5 minutes. User B can check to see what the new global rate is if MONITOR SESSION is being run. The data returned in the first Record parcel is the SampleSec, which indicates the duration of the collection period in seconds. User B can observe that SampleSec reports 300 seconds (or 5 minutes) instead of 600 seconds (or 10 minutes).                                                                                                                                                                                      |

## Rate Scenario 2: Effects On Other Users When Increasing the Local Rate

Users A and B are requesting session-level data at the same rate as data is collected.

| Step | Action                                                                                                                                                                                                                                                                        |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | User A wants to troubleshoot a new application program and at 9:00 sets a local rate of 1 minute within his session. The local rate now overrides the global rate of 10 minutes within the session for User A. User A now requests data every 1 minute instead of 10 minutes. |



| Step | Action                                                                                                                                                                                                                           |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2    | User A makes a request at 9:01. Data returned shows data from the interval of 9:00 to 9:01 (because User A has reset the session beginning collection time to 9:00), plus the cumulative data from 8:00 to 9:00.                 |
| 3    | User A makes the next request at 9:02. Data returned shows data from the interval of 9:01 to 9:02, plus the cumulative data from 8:00 to 9:01. From now on, responses to queries for User A returns data updated every 1 minute. |

The actions made by User A in setting a local rate does not change the current global collection rate of 10 minutes.

The way the session for User B is calculating its collection interval does not change. Therefore, the collection rate and request rate for User B continues at 10-minute intervals. However, since all session usage data is stored in the same memory repository, the more frequent collection for User A may affect the cumulative data that User B might see.

User B may see more dynamic data than previously and receive more data than expected. Each of the requests made by User B could potentially include data from one collection period mixed in with data from a subsequent collection period that was initiated by a different use. In this case, the 1-minute interval collection for User A.

While the local rate for User A is in effect, User B makes a request at 9:00. Data returned shows data from the interval of 8:50 to 9:00, plus the cumulative data from 8:00 to 8:50.

If User B were to make a request at 9:04, User B would see data returned from 9:00 to 9:04, plus the cumulative data from 8:00 to 9:00. Because of the local data collecting for User A, User B can take advantage of the faster collection rate and thus receive more data than expected.

On the other hand, if the local rate for User A were not in effect and User B makes a request at 9:04, the data returned would be from the interval of 8:00 to 9:00 only.

### Rate Scenario 3: How Request Order Affects Data Reported

The DBA sets the global rate at 8:00 a.m. to collect every 10 minutes. User A decides to reset the global or global rate from 10 minutes to 5 minutes. In addition, User B decides to request session-level data at a different rate than the data collection rate.

| Step | Action                                                                                                                                           |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | User A resets the global session collection rate to 300 seconds (or 5 minutes) at 9:00, thus setting the timestamp for any current data to 9:00. |
| 2    | At 9:00, User A also sets a local collection rate of 1 minute for his session.                                                                   |
| 3    | User B makes a request for data at 9:05:25.                                                                                                      |
| 4    | User B receives the cumulative data from 8:00 to 9:05:25 and, in the process, resets the data time-stamp to 9:05:25.                             |

User A does not know that the data timestamp is altered by the request made by User B because User A had sent in a request immediately after the request made by User B.

The request made by User A for data at 9:06:00 is queued behind the request made by User B of 9:05:25. Because the current data is considered current (9:06:00 - 9:05:25 = 0:00:35, which is less than the local collection rate of 1 minute), User A receives the same data (from 8:00 to 9:05:25) that User B received.

In this case, the request order affects the data User A receives. If not the earlier request made by User B, User A could have received data from the 8:00 to 9:06 interval, which User A probably expected.

### Rate Scenario 4: Different Collection and Request Rates

Users A and B are running different application programs and are monitoring the applications with MONITOR SESSION and/or MONITOR VIRTUAL SUMMARY. User A has the SET SESSION RATE privilege. User A is requesting session-level data at a different rate from the rate at which data is collected.

User A is using the global collection rate of 10 minutes set by the DBA. However, User A wants to troubleshoot a new program and decides to set the application program to request data every 6 seconds.

Clearly, User A is requesting data more frequently than the data collection rate. Every 6 seconds, User A sees a data display that will show the same data until the 10-minute interval has elapsed.

In this case, data returned from monitoring the session for User A would probably be more meaningful if User A had set the global collection rate to the same rate as the request rate.

### CLv2 Response Parcels

The response returned from the the database contains the following sequence of parcel type:

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                                                  |
|-----------------|---------------|----------------|----------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273      | Activity Count = Previous rate<br>Activity Type = 83 (PCLSETSESSR)               |
| DataInfo        | 71            | 6 to 64100     | This parcel is present if request was IndicReq parcel; depends on the data type. |
| EndStatement    | 11            | 6              | StatementNo: 2-byte integer                                                      |
| EndRequest      | 12            | 4              | None                                                                             |

### Sample Input - CLv2 Request

This example shows how the parcels for a SET SESSION RATE request, built by CLv2, look when sent to the database server with a *sample\_rate* of 6 and a *local\_change* of y. In the example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size. However, a minimum response size is 32,000 bytes.

| Flavor |      | Length | Body                               |                  |
|--------|------|--------|------------------------------------|------------------|
| Num    | Name | Bytes  | Field                              | Value            |
| 0001   | Req  | 20     | Request                            | SET SESSION RATE |
| 0003   | Data | 9      | MonVerID<br>SampleRate<br>LocalChg | 2<br>6<br>Y      |
| 0004   | Resp | 6      | BufferSize                         | 64000            |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

With a *sample\_rate* of 6 and a *local\_change* of y, this example shows the values returned in character text format for the SET SESSION RATE request. Your application program may display returned values in a different format.

```
Success parcel:
  StatementNo: 1      ActivityCount: 0
  ActivityType: 83    FieldCount: 0
DataInfo parcel:
  FieldCount: 0
EndStatement.
EndRequest.
```

### Relationship Between SET SESSION RATE and MONITOR SESSION

You must execute the SET SESSION RATE request to activate session-level data collection before you execute a MONITOR SESSION request. This means that either the global rate or local rate must be set to nonzero. If both rates are set to zero, an error message is returned.

For information on this PMPC PM/API request relationship, see [Relationship Between MONITOR SESSION and SET SESSION RATE](#).

### Relationship Between SET SESSION RATE and MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY

Changes to the session-level rates (global and local) specified by SET SESSION RATE are reported in the data returned by MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY.

**Note:**

The local rate reported is your own local rate. If the local rate is not set, the local rate is reported as zero.

As more session-level monitoring is done (by setting a faster SET SESSION RATE), the resulting overhead may increase the level of CPU usage (reported in MONITOR PHYSICAL SUMMARY or MONITOR VIRTUAL SUMMARY data) by your system. However, this may depend on the size of the rate change and the type of work done by other sessions.

## Open APIs (SQL Interfaces)

This section describes how to access PMPC data through SQL interfaces consisting of UDFs. The PMPC UDFs, like the PMPC PM/API requests, are used to monitor system and session-level performance and set collection and logging rates. However, the PMPC UDFs can be invoked from any application.

The PMPC UDFs provide similar functionality to the PMPC PM/API requests.

When issuing PMPC UDFs, you do not need to fully qualify them by the database name, SYSLIB. They are automatically searched by the DBS in the SYSLIB database.

## AbortListSessions

Aborts the session and lists the status of the aborted sessions.

### Syntax

```
REPLACE FUNCTION SYSLIB.AbortListSessions (
  HostIdIn SMALLINT,
  UserNameIn TD_ANYTYPE,
  SessionNoIn INTEGER,
  LogoffSessions VARCHAR(1) CHARACTER SET LATIN,
  UserOverride VARCHAR(1) CHARACTER SET LATIN
) RETURNS TABLE (
  HostId SMALLINT,
  SessionNo INTEGER,
  UserName VARCHAR(128) CHARACTER SET UNICODE,
  AbortStatus CHAR CHARACTER SET LATIN
)
...
;
```

## Syntax Elements

### *HostIdIn*

Logical ID of a host (or client) with sessions logged on. A value of -1 indicates all hosts.

### *UserNameIn*

User name of the session. An asterisk (\*) or NULL indicates all users.

### *SessionNoIn*

ID of the session to abort. A value of zero indicates all sessions.

### *LogoffSessions*

Indicator of whether to log off sessions to Vantage in addition to aborting them:

- Y = Log off and abort sessions.
- N or NULL = Do not log sessions off.

### *UserOverride*

Indicator of whether to override an ABORT SESSION failure:

- Y = Override the ABORT SESSION request to fail in any of the following cases:
  - An identified session is being session-switched.
  - An identified session is executing its own ABORT SESSION request.
  - An identified session has a PESTate of IDLE: IN-DOUBT as a result of a 2PC.

Sessions are marked IN-DOUBT by the 2PC protocol, which governs how transactions are committed by multiple systems that do not share memory. The protocol guarantees that either all systems commit or all roll back.

- N or NULL = Do not override.

### *HostId*

Logical host ID of (or client) the aborted sessions were logged on to. For a PE, HostId identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs. A value of zero represents the Supervisor window.

### *SessionNo*

Number of the session that was aborted. Together with a given host ID, a session number uniquely identifies a session on the Teradata system. This value is assigned by the host (or client) at logon time.

**UserName**

User name of the session that was aborted.

**AbortStatus**

Status of the sessions aborted:

- I = In-Doubt
- A = Aborting a transaction
- C = Committing a transaction
- E = Executing an ABORT SESSION request
- S = Switching
- An empty string = In some state other than the above. This value returns when issuing an SQL function.

For an ABORT without LOGOFF, any status except NULL indicates the reason the request could not impact the associated session.

For an ABORT with LOGOFF, an I, E, or S status value indicates that the associated session cannot be aborted or logged off.

**Usage Notes**

This table function is only supported in Constant Mode.

AbortListSessions cannot be used to abort delayed utility sessions because these sessions are not completely logged on.

This function provides similar functionality to the PMPC ABORT SESSION request with ListSessions set to 'N' or 'No'.

**Example: Using AbortListSessions**

```
SELECT * FROM TABLE (AbortListSessions(1, 'User14', 0, 'Y', 'Y')) AS t1;
```

```
*** Query completed. 5 rows found. 4 columns returned.
```

```
*** Total elapsed time was 4 seconds.
```

| HostId | SessionNo | UserName | AbortStatus |
|--------|-----------|----------|-------------|
| 1      | 1007      | USER14   |             |
| 1      | 1011      | USER14   |             |
| 1      | 1010      | USER14   |             |
| 1      | 1009      | USER14   |             |
| 1      | 1008      | USER14   |             |

## AbortSessions

Aborts any outstanding request or transaction of one or more sessions, and optionally logs those sessions off the database.

If successful, this function returns the number of sessions that were aborted.

### Syntax

```
REPLACE FUNCTION SYSLIB.AbortSessions (
  HostIdIn SMALLINT,
  UserNameIn TD_ANYTYPE,
  SessionNoIn INTEGER,
  LogoffSessions VARCHAR(1) CHARACTER SET LATIN,
  UserOverride VARCHAR(1) CHARACTER SET LATIN
) RETURNS INTEGER
...
;
```

### Syntax Elements

#### *HostIdIn*

Logical ID of a host (or client) with sessions logged on. A value of -1 indicates all hosts.

#### *UserNameIn*

User name of the session. An asterisk (\*) or NULL indicates all users.

#### *SessionNoIn*

ID of the session to abort. A value of zero indicates all sessions.

#### *LogoffSessions*

Indicator of whether to log off sessions to Vantage in addition to aborting them:

- Y = Log off and abort sessions.
- N or NULL = Do not log sessions off.

#### *UserOverride*

Indicator of whether to override an ABORT SESSION failure:

- Y = Override the ABORT SESSION request to fail in any of the following cases:
  - An identified session is being session-switched.
  - An identified session is executing its own ABORT SESSION request.
  - An identified session has a PESTate of IDLE: IN-DOUBT as a result of a 2PC.

Sessions are marked IN-DOUBT by the 2PC protocol, which governs how transactions are committed by multiple systems that do not share memory. The protocol guarantees that either all systems commit or all roll back.

- N or NULL = Do not override.

## Usage Notes

The AbortSessions function provides similar functionality to the PMPC ABORT SESSION request with ListSessions set to 'Y' or 'Yes'. For information about this interface, see [ABORT SESSION](#).

### Example: Using AbortSessions

```
SELECT AbortSessions (1, 'User14', 0, 'Y', 'Y');
* * * Query completed. One row found. one column returned.
* * * Total elapsed time was 5 seconds.
AbortSessions (1, 'User14', 0, 'Y', 'Y')
-----
5
```

### Example: Using AbortSessions with MonitorSession

```
SELECT AbortSessions (HostId, UserName, SessionNo, 'Y', 'Y');
      FROM TABLE (MonitorSession(-1, '*', 0)) AS t1
WHERE username= 'user14';
* * * Query completed. 5 rows found. one column returned.
* * * Total elapsed time was 2 seconds.
AbortSessions (HostId, UserName, SessionNo, 'Y', 'Y')
-----
1
1
1
1
```

## IdentifyDatabase

Returns the name of the specified database ID.

### Syntax

```
REPLACE FUNCTION SYSLIB.IdentifyDatabase (
  Databaseld INTEGER
) RETURNS VARCHAR(128) CHARACTER SET UNICODE
```



```
...
;
```

## Syntax Elements

### *DatabaseId*

Database ID.

## Usage Notes

The IdentifyDatabase function provides similar functionality to the PMPC IDENTIFY DATABASE request. For information about this interface, see [IDENTIFY](#).

## Example: Using IdentifyDatabase

```
SELECT IdentifyUser(blk1userid) as "blocking user",
       IdentifyTable(blk1objtid) as "blocking table",
       IdentifyDatabase(blk1objdbid) as "blocking db"
FROM TABLE (MonitorSession(-1,'*',0)) AS t1
WHERE Blk1UserId > 0;
*** Query completed. One row found. 3 columns returned.
*** Total elapsed time was 4 seconds.
```

| blocking user | blocking table | blocking db |
|---------------|----------------|-------------|
| -----         | -----          | -----       |
| user1         | skewAmp1       | DBaaa       |

## IdentifySession

Returns the name of the user, by session, who is causing a block.

## Syntax

```
REPLACE FUNCTION SYSLIB.IdentifySession (
  HostId INTEGER,
  SessionNo INTEGER
) RETURNS VARCHAR(128) CHARACTER SET UNICODE
...
;
```

## Syntax Elements

### *HostId*

ID of the blocking host.

**SessionNo**

Number of the session that is blocked.

**Usage Notes**

The IdentifySession function provides similar functionality to the PMPC IDENTIFY SESSION request. For information about this interface, see [IDENTIFY](#).

**Example: Using IdentifySession**

```
SELECT IdentifySession(blk1Hostid,blk1sessno)
FROM TABLE (MonitorSession(-1,'*',0)) AS t1
WHERE Blk1UserId > 0;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
IdentifySession(Blk1HostId,Blk1SessNo)
-----
USER1
```

**IdentifyTable**

Returns the name of the specified table ID.

**Syntax**

```
REPLACE FUNCTION SYSLIB.IdentifyTable (
  TableID  INTEGER
) RETURNS VARCHAR(128) CHARACTER SET UNICODE
  ...
;
```

**Syntax Elements****TableID**

ID of the locked table.

**Usage Notes**

The IdentifyTable function provides similar functionality to the PMPC IDENTIFY TABLE request. For information about this interface, see [IDENTIFY](#).

**Example: Using IdentifyTable**

```

SELECT IdentifyUser(blk1userid) as "blocking user",
       IdentifyTable(blk1objtid) as "blocking table",
       IdentifyDatabase(blk1objdbid) as "blocking db"
FROM TABLE (MonitorSession(-1,'*',0)) AS t1
WHERE Blk1UserId > 0;
*** Query completed. One row found. 3 columns returned.
*** Total elapsed time was 4 seconds.
blocking user           blocking table           blocking db
-----
user1                   skewAmp1                 DBaaa

```

**IdentifyUser**

Returns the name of the specified user ID who is causing a block.

**Syntax**

```

REPLACE FUNCTION SYSLIB.IdentifyUser (
  UserId  INTEGER
) RETURNS VARCHAR(128) CHARACTER SET UNICODE
  ...
;

```

**Syntax Elements*****UserId***

ID of the user who is causing the block.

**Usage Notes**

The IdentifyUser function provides similar functionality to the PMPC IDENTIFY USER request. For information about this interface, see [IDENTIFY](#).

**Example: Using IdentifyUser**

```

SELECT IdentifyUser(blk1userid) as "blocking user",
       IdentifyTable(blk1objtid) as "blocking table",
       IdentifyDatabase(blk1objdbid) as "blocking db"
FROM TABLE (MonitorSession(-1,'*',0)) AS t1
WHERE Blk1UserId > 0;
*** Query completed. One row found. 3 columns returned.
*** Total elapsed time was 4 seconds.

```

| blocking user | blocking table | blocking db |
|---------------|----------------|-------------|
| -----         | -----          | -----       |
| user1         | skewAmp1       | DBaaa       |

## MonitorAMPLoad

Returns the data from the third statement of the MONITOR AWT RESOURCE request response.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorAMPLoad (
) RETURNS TABLE (
    VprocNo SMALLINT,
    AvailableAWTs INTEGER,
    InUseAWTs INTEGER,
    MsgCount INTEGER,
    DQMsgCount INTEGER
    AvailableAWTsForAll INTEGER
)
...
;

```

### Syntax Elements

#### VprocNo

AMP vproc number.

#### AvailableAWTs

Number of available AMP worker tasks in each AMP.

#### InUseAWTs

Number of active AMP worker tasks in each AMP.

#### MsgCount

Number of messages currently queued for delivery to each AMP.

#### DQMsgCount

Number of messages processed by each AMP.

#### AvailableAWTsForAll

The number of available AMP worker tasks in the unreserved pool in each AMP.

## Usage Notes

For information on the third statement returned from the MONITOR AWT RESOURCE request response, see [Statement 3 - MONITOR AWT RESOURCE](#).

### Example: Data Reported in Statement 3 of MONITOR AWT RESOURCE

This example shows the data reported in statement 3 of MONITOR AWT RESOURCE.

```
SELECT * FROM TABLE (MonitorAMPLoad()) AS t1;
```

```
*** Query completed. 4 rows found. 6 columns returned.
```

```
*** Total elapsed time was 1 second.
```

|                     |     |
|---------------------|-----|
| VprocNo             | 0   |
| AvailableAWTs       | 49  |
| InUseAWTs           | 2   |
| MsgCount            | 0   |
| DQMsgCount          | 737 |
| AvailableAWTsForAll | 55  |
| VprocNo             | 1   |
| AvailableAWTs       | 49  |
| InUseAWTs           | 1   |
| MsgCount            | 0   |
| DQMsgCount          | 765 |
| AvailableAWTsForAll | 56  |
| VprocNo             | 2   |
| AvailableAWTs       | 49  |
| InUseAWTs           | 1   |
| MsgCount            | 0   |
| DQMsgCount          | 695 |
| AvailableAWTsForAll | 56  |
| VprocNo             | 3   |
| AvailableAWTs       | 48  |
| InUseAWTs           | 2   |
| MsgCount            | 0   |
| DQMsgCount          | 863 |
| AvailableAWTsForAll | 56  |

## MonitorAWTResource

Collects statistics on AMPs based on the in-use AMP Worker Tasks (AWTs).

**Syntax**

```

REPLACE FUNCTION SYSLIB.MonitorAWTResource (
  Threshold1 INTEGER,
  Threshold2 INTEGER,
  Threshold3 INTEGER,
  Threshold4 INTEGER
) RETURNS TABLE (
  IntervalCount1 INTEGER,
  IntervalCount2 INTEGER,
  IntervalCount3 INTEGER,
  IntervalCount4 INTEGER,
  FlowControl INTEGER,
  HighAMP1VprocId INTEGER,
  HighAMP1InUseCount INTEGER,
  HighAMP2VprocId INTEGER,
  HighAMP2InUseCount INTEGER,
  HighAMP3VprocId INTEGER,
  HighAMP3InUseCount INTEGER,
  HighAMP4VprocId INTEGER,
  HighAMP4InUseCount INTEGER,
  HighAMP5VprocId INTEGER,
  HighAMP5InUseCount INTEGER,
  LowAMP1VprocId INTEGER,
  LowAMP1InUseCount INTEGER,
  LowAMP2VprocId INTEGER,
  LowAMP2InUseCount INTEGER,
  LowAMP3VprocId INTEGER,
  LowAMP3InUseCount INTEGER,
  LowAMP4VprocId INTEGER,
  LowAMP4InUseCount INTEGER,
  LowAMP5VprocId INTEGER,
  LowAMP5InUseCount INTEGER,
  FCVprocId1 INTEGER,
  FCVprocId2 INTEGER,
  FCVprocId3 INTEGER,
  FCVprocId4 INTEGER,
  FCVprocId5 INTEGER
)
...
;

```

## Syntax Elements

### *Threshold1*

Minimum value for in-use AWTs to qualify an AMP for inclusion into this interval.

### *Threshold2*

### *Threshold3*

### *Threshold4*

Start value for in-use AWTs to qualify an AMP for inclusion into this interval.

### *IntervalCount $n$*

Where  $n$  in [1, 4]: Number of AMPs in the system whose in-use AWT counts fall at, or above, the *Threshold $n$*  value and do not qualify for the higher thresholds.

### *FlowControl*

Number of AMPs currently in some form of Flow Control.

### *HighAMP $n$ VprocId*

Where  $n$  in [1, 5]: Vproc ID of the AMP with the  $n$ th highest in-use count in the system. A value of -1 indicates this field is not defined.

### *HighAMP $n$ InUseCount*

Where  $n$  in [1, 5]: In-use count associated with *HighAMP $n$ VprocId*. Applicable only if *HighAMP $n$ VprocId* is defined.

### *LowAMP $n$ VprocId*

Where  $n$  in [1, 5]: Vproc ID of the AMP with the  $n$ th lowest in-use count in the system. A value of -1 indicates this field is not defined.

### *LowAMP $n$ InUseCount*

Where  $n$  in [1, 5]: In-use count associated with *LowAMP $n$ VprocId*. Applicable only if *LowAMP $n$ VprocId* is defined.

### *FCVprocId $n$*

Where  $n$  in [1, 5]: The vproc ID of the AMP in flow control. A value of -1 indicates this field is not defined.

## Usage Notes

The *MonitorAWTRResource* functions are two overloaded functions. One requires the threshold values to be passed in; the other sets the threshold values to 45, 55, 62 and 75.

The MonitorAWTResource function provides similar functionality to the PMPC MONITOR AWT RESOURCE request. For information about this interface, see [MONITOR AWT RESOURCE](#).

### Example: Using MonitorAWTResource

```
SELECT * FROM TABLE (MonitorAWTResource(1,2,3,4)) AS t1;
*** Query completed. One row found. 30 columns returned.
*** Total elapsed time was 1 second.
```

|                    |    |
|--------------------|----|
| IntervalCount1     | 0  |
| IntervalCount2     | 0  |
| IntervalCount3     | 2  |
| IntervalCount4     | 4  |
| FlowControl        | 0  |
| HighAMP1VprocId    | 2  |
| HighAMP1InUseCount | 6  |
| HighAMP2VprocId    | 5  |
| HighAMP2InUseCount | 4  |
| HighAMP3VprocId    | 3  |
| HighAMP3InUseCount | 4  |
| HighAMP4VprocId    | 1  |
| HighAMP4InUseCount | 4  |
| HighAMP5VprocId    | 4  |
| HighAMP5InUseCount | 3  |
| LowAMP1VprocId     | 4  |
| LowAMP1InUseCount  | 3  |
| LowAMP2VprocId     | 0  |
| LowAMP2InUseCount  | 3  |
| LowAMP3VprocId     | 5  |
| LowAMP3InUseCount  | 4  |
| LowAMP4VprocId     | 3  |
| LowAMP4InUseCount  | 4  |
| LowAMP5VprocId     | 1  |
| LowAMP5InUseCount  | 4  |
| FCVprocId1         | -1 |
| FCVprocId2         | -1 |
| FCVprocId3         | -1 |
| FCVprocId4         | -1 |
| FCVprocId5         | -1 |

## MonitorMySessions

Collects the session information for the current user on the current host.



**Syntax**

```

REPLACE FUNCTION SYSLIB.MonitorMySessions (
) RETURNS TABLE (
    HostId SMALLINT,
    SessionNo INTEGER,
    LogonPENO SMALLINT,
    RunVprocNo SMALLINT,
    PartName VARCHAR(16) CHARACTER SET LATIN,
    PEstate VARCHAR(18) CHARACTER SET LATIN,
    LogonTime VARCHAR(22) CHARACTER SET LATIN,
    UserId INTEGER,
    LSN INTEGER,
    UserName VARCHAR(128),
    UserAccount VARCHAR(128),
    PECPUsec FLOAT,
    XActCount FLOAT,
    ReqCount FLOAT,
    ReqCacheHits FLOAT,
    AMPState VARCHAR(18) CHARACTER SET LATIN,
    AMPCPUsec FLOAT,
    AMPIO FLOAT,
    ReqSpool FLOAT,
    Blk1HostId SMALLINT,
    Blk1SessNo INTEGER,
    Blk1UserId INTEGER,
    Blk1Lmode CHAR(1) CHARACTER SET LATIN,
    Blk1Otype CHAR (2) CHARACTER SET LATIN,
    Blk1ObjDBID INTEGER,
    Blk1ObjTID INTEGER,
    Blk1Status CHAR (1) CHARACTER SET LATIN,
    Blk2HostId SMALLINT,
    Blk2SessNo INTEGER,
    Blk2UserId INTEGER,
    Blk2Lmode CHAR(1) CHARACTER SET LATIN,
    Blk2Otype CHAR(2) CHARACTER SET LATIN,
    Blk2ObjDBID INTEGER,
    Blk2ObjTID INTEGER,
    Blk2Status CHAR(1) CHARACTER SET LATIN,
    Blk3HostId SMALLINT,
    Blk3SessNo INTEGER,
    Blk3UserId INTEGER,
    Blk3Lmode CHAR(1) CHARACTER SET LATIN,

```

```

Blk30type CHAR(2) CHARACTER SET LATIN,
Blk30objDBID INTEGER,
Blk30objTID INTEGER,
Blk3Status CHAR(1) CHARACTER SET LATIN,
MoreBlockers CHAR(1) CHARACTER SET LATIN,
LogonSource VARCHAR(128) CHARACTER SET UNICODE,
HotAmp1CPU FLOAT,
HotAmp2CPU FLOAT,
HotAmp3CPU FLOAT,
HotAmp1CPUId FLOAT,
HotAmp2CPUId FLOAT,
HotAmp3CPUId FLOAT,
HotAmp1IO FLOAT,
HotAmp2IO FLOAT,
HotAmp3IO FLOAT,
HotAmp1IOId INTEGER,
HotAmp2IOId INTEGER,
HotAmp3IOId INTEGER,
LowAmp1CPU FLOAT,
LowAmp2CPU FLOAT,
LowAmp3CPU FLOAT,
LowAmp1CPUId INTEGER,
LowAmp2CPUId INTEGER,
LowAmp3CPUId INTEGER,
LowAmp1IO FLOAT,
LowAmp2IO FLOAT,
LowAmp3IO FLOAT,
LowAmp1IOId INTEGER,
LowAmp2IOId INTEGER,
LowAmp3IOId INTEGER,
AvgAmpCPUSec FLOAT,
AvgAmpIOCnt FLOAT,
AmpCount SMALLINT,
TempSpaceUsg FLOAT,
ReqStartTime VARCHAR(22) CHARACTER SET LATIN,
ReqCPU FLOAT,
ReqIO FLOAT,
ReqNo INTEGER,
WlcId INTEGER,
DontReclassifyFlag SMALLINT,
ProxyUser VARCHAR (128) CHARACTER SET UNICODE,
CPUDecayLevel SMALLINT,
IODecayLevel SMALLINT,
TacticalCPUException INTEGER,

```

```

TacticalIOException INTEGER,
ReqIOKB FLOAT,
ReqPhysIO FLOAT,
ReqPhysIOKB FLOAT,
ReqStepsCompletedCnt INTEGER,
RedriveProtection CHAR (2),
CurrentRedriveParticipation CHAR (1),
ReqRedriveSpoolSpace FLOAT,
BlockerSessionCnt SMALLINT,
ReqTblOpBytesIn FLOAT,
ReqTblOpBytesOut FLOAT,
ProxyUserId INTEGER,
ZoneId INTEGER,
ReqHotAmpCPU FLOAT,
ReqHotAmpCPUId SMALLINT,
ReqHotAmpIO FLOAT,
ReqHotAmpIOId SMALLINT,
ReqInvolvedAMPCnt SMALLINT,
ReqFirstRespTime VARCHAR(22) CHARACTER SET LATIN,
ReqLocalQueryStatus SMALLINT,
ReqRemoteHostId SMALLINT,
ReqRemoteSessionId INTEGER,
ReqRemoteRequestId INTEGER,
ReqRemoteQueryId FLOAT
ReqHotAmpSpool FLOAT,
ReqHotAmpSpoolId SMALLINT,
ReqMapNo SMALLINT,
ReqMaxNumMapAMPs INTEGER,
ReqMinNumMapAMPs INTEGER,
ReqSysDefNumMapAMPs INTEGER,
ReqRemoteHostIp VARCHAR(128) CHARACTER SET UNICODE,
ReqServerName VARCHAR(128) CHARACTER SET UNICODE,
ReqFlexReleased SMALLINT,
ReqAdmissionTime VARCHAR(22) CHARACTER SET LATIN
)
...
;

```

## Syntax Elements

### HostId

Logical Host ID associated with a PE or session.

For a PE, HostId identifies one of the hosts or LANs associated with the described PE.

For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs. A value of zero represents the Supervisor window.

### **SessionNo**

Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.

### **LogonPENo**

Vproc number of the PE the session is currently logged on to; it identifies the PE that has control responsibility for the session. Normally, this is the PE that processed the logon request; but if that PE goes offline, it will be the PE to which the session was switched.

### **RunVprocNo**

Vproc number of the AMP or PE currently assigned to process the session requests.

For sessions in Teradata SQL partitions, this value is identical to the LogonPENo. For sessions in FastLoad or MultiLoad partitions, this is the AMP that initially processes the data being loaded.

If a RunVprocNo value of -1 in record mode or NULL in indicator mode is returned by MonitorMySessions for FastLoad, MultiLoad or FastExport sessions, this indicates that the session is in the process of starting up.

### **PartName**

Name of the session partition associated with this session. Following a successful logon request by a session or as part of a connect request, the session identifies the partition with which the user wants the session to be associated. FASTLOAD, Teradata SQL, MONITOR, INTERNAL are examples of valid partition names.

### **PEState**

Current state of the session within the PE. See the MONITOR SESSION PESTate field for a list of the PESTate values.

### **LogonTime**

Date and time portion of the information recorded by Session Control when a session successfully logs on. It is usually formatted for display as *yyyy/mm/dd 99:99:99.99*, which represents the year, month, day, hours: minutes: seconds.

**UserId**

User or internal ID of a user for this session. Within the database, UserId is equivalent to Database ID. Typically, UserId is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, UserId can identify either a given user or database name.

**LSN**

Logon Sequence Number (LSN) that is associated with a session when the session logs on. It identifies a collection of sessions performing a related activity. For example, in a FastLoad job, a user is logged on as a Teradata SQL session, as well as  $n$  FastLoad sessions with the same user name. Therefore,  $n+1$  sessions (1 Teradata SQL and  $n$  FastLoad) with the same LSN are all associated with the given FastLoad job. To see how the FastLoad job is doing, the user can pick out all sessions reported with the same LSN number.

This information supplies the parent-child relationship for sessions involved with FastLoad and MultiLoad jobs.

**UserName**

User name of the session.

**UserAccount**

Current account for the session.

**PECPUseC**

CPU time (in seconds) used in a PE by the associated session for parsing and dispatching requests. It is accurate to the second.

This value is valid only when associated with Teradata SQL and MONITOR partition sessions.

**XActCount**

Number of explicit and implicit transactions executed by the session.

This value is valid only when returned for Teradata SQL sessions, and is NULL for all other partition sessions. For this value, you must make a request for data before completion of the first collection period that follows either a system outage or a change in the ResMonitor rate.

**ReqCount**

Number of requests (Tequel Start Request [TSR] messages) initiated by the session.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or change in the ResMonitor rate.

**ReqCacheHits**

Number of times that this session processed a request using information from the Teradata SQL Parser request cache, specifically, how many times there was a request cache hit.

This value is valid only for Teradata SQL sessions, and is NULL for all other partition sessions. This value is also NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

**AMPState**

Current state of the associated session in AMP vprocs in decreasing priority:

| State    | Description                                                                                   |
|----------|-----------------------------------------------------------------------------------------------|
| ABORTING | Transaction being rolled back; session aborting.                                              |
| BLOCKED  | Background activity in progress; last request on hold until background activity is completed. |
| ACTIVE   | Normal, on-going activity.                                                                    |
| IDLE     | No work in progress on any AMP.                                                               |
| UNKNOWN  | No recorded activity since monitoring began.                                                  |

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**AMPCPUSec**

Current elapsed CPU time, in seconds, used on all AMPs by the associated session for executing requests. For example, for Teradata SQL requests, this is the time spent by the database actively working or rolling back an aborted transaction. This does not include any PDE CPU time spent handling database requests.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

**AMPIO**

Current number of logical Reads and Writes issued across all AMPs by the associated session.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

**ReqSpool**

Current spool used by current request across all AMPs, expressed as a number of bytes.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

**Blk1HostId****Blk2HostId****Blk3HostId**

Logical host ID of a session causing a block. This value is derived from equating the transactions causing a database lock conflict to the sessions that issued those transactions. The Blk\_x\_HostId in combination with Blk\_x\_SessNo uniquely identifies the session that is causing a block.

This value is NULL if:

- The host ID is not available.
- The session does not have an AMPState of BLOCKED.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserId values all return as NULLs and AMPState is BLOCKED, a Host Utility (HUT) lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1SessNo****Blk2SessNo****Blk3SessNo**

Number of the session causing a block. This value is derived from associating the transactions causing a lock conflict to the sessions that issued those transactions. The Blk\_x\_SessNo in combination with Blk\_x\_HostId uniquely identifies the session causing a block.

This value is NULL if:

- SessNo is not available.
- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserId values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1UserId****Blk2UserId****Blk3UserId**

ID of the user or host utility job preventing the session from being granted a lock. The user ID is the only information available about who placed the lock.

This value is NULL if:

- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1Lmode****Blk2Lmode****Blk3Lmode**

Mode (severity) of the lock involved in causing a block:

- E = Exclusive
- W = Write
- R = Read
- A = Access

Locks are reported in decreasing order of severity because removing the most severe lock conflict may eliminate the source of the lock conflict.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

A session may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock in the queue and is in conflict with the lock requested by this blocked session. For information on whether the lock is granted, see the MONITOR SESSION fields: Blk1Status, Blk2Status, and Blk3Status.



**Blk1Otype****Blk2Otype****Blk3Otype**

Type of object whose lock is causing the session described by the associated row to be blocked:

- D = Database
- T= Table
- R = Row hash
- TP = Table Partition range
- RP = RowHash in Partition range
- RK = RowHash in one partition
- RN = RowKey range

However, this object is not necessarily the type of object the blocked session is trying to access. For example, if the session is requesting a row hash lock, the blocking object could be a database, table, or row hash.

This value is NULL if:

- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

For a Table T, it is possible for User A to block User B with a table level lock on Table T on AMP\_1 and with a Row Hash Level lock on that same Table T on AMP\_2. When that occurs, the only lock conflict reported is that User B is blocked by User A on a table.

**Blk1ObjDBID****Blk2ObjDBID****Blk3ObjDBID**

Unique ID of the database object over which a lock conflict is preventing the session from being granted a lock.

Within the database system, Database ID is equivalent to User ID. Typically, User ID is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, Database ID can identify either a user or database name.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**Blk1ObjTID****Blk2ObjTID****Blk3ObjTID**

Unique ID of the table object over which a lock conflict is preventing the session from being granted a lock.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- The Blk\_x\_OType is D.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**Blk1Status****Blk2Status****Blk3Status**

Status of lock causing a block:

- W= Waiting
- G = Granted

This value is NULL if:

- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

A lock request may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock request in the queue and is in conflict with it.

A status of Waiting has a higher priority than that of Granted when there is more than one lock involved. For example, for a given object and a given session, a session that is blocked by a Waiting lock on one AMP and a Granted lock on another AMP has Waiting reported as its status.

**MoreBlockers**

Indicator of more lock conflicts:

- Blank = Blk x information is a complete list of sessions blocking the session described.
- Asterisk (\*) = Additional sessions are blocking the session described.

This value is NULL if:

- The state of the session is not BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**LogonSource**

Logon source information. At logon time, this information is optionally supplied by the Teradata Director Program or the Gateway to further identify the physical or logical location of the session, the logon user name, and the interface under which the session was initiated. (For example, the data string may include DBC as the user ID and BTEQ as the interface.)

A two-byte value precedes the LogonSource data to indicate the length of the string. The length value is zero if LogonSource is NULL.

For a list of the commonly seen LogonSource string application names, see *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

**HotAmpnCPU**

Where  $n$  in  $[1, 3]$ : CPU time of the  $n$ th highest CPU utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

**HotAmpnCPUId**

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th highest CPU utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

**HotAmpnIO**

Where  $n$  in  $[1, 3]$ : I/O count of the  $n$ th highest I/O utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

**HotAmpnIOld**

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th highest I/O utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **LowAmpnCPU**

Where  $n$  in  $[1, 3]$ : CPU time of the  $n$ th lowest CPU utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **LowAmpnCPUId**

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th lowest CPU utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **LowAmpnIO**

Where  $n$  in  $[1, 3]$ : I/O count of the  $n$ th lowest I/O utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **LowAmpnIOId**

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th lowest I/O utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **AvgAmpCPUsec**

Average AMP CPU utilization for the last session collection interval. The average is calculated as the sum of CPU utilization for all amps participating divided by the number of online AMPs.

This value is NULL if the request is made before the collection period expires.

### **AvgAmpIOCnt**

Average AMP I/O utilization for the last session collection interval. The average is calculated as the sum of I/O utilization for all AMPs participating divided by the number of online AMPs.

This value is NULL if the request is made before the collection period expires.

### **AmpCount**

Current number of AMPs currently executing on the associated node.

### **TempSpaceUsg**

Total amount, in bytes, of temporary space used by the session.

This value is NULL if the session did not materialize any temporary tables.

### **ReqStartTime**

Date and time of the current request on the session started. It is usually formatted for display as *yyyy/mm/dd 99:99:99.99*, which represents the year, month, day hours: minutes: seconds.

### **ReqCPU**

Total CPU usage by the current SQL request on the session on all AMPs. This value contains proper request-level statistics for DBC/SQL sessions running SQL requests only. Ignore the value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.

This field is equivalent to the MONITOR SESSION RequestAmpCPU field.

### **ReqIO**

Total number of accesses by the current SQL request for the session on all AMPs. This value contains proper request-level statistics for DBC/SQL sessions running SQL requests only. Ignore the value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.

This field is equivalent to the MONITOR SESSION RequestAmpI/O field.

In some rare cases, in the very early phase of a request in parsing state, when PESTate = PARSING, the request number may not be available and returned as zero or NULL. The active request number will be available on the next collection.

### **WlId**

Workload ID associated with the specified request.

**DontReclassifyFlag**

Flag indicating that the next request on the session will not be classified but will use the workload ID (WlId) already assigned to the session. This will occur if this is a utility session or a WlId was assigned to the session using the TDWMAssignWD function or the TDWM WD ASSIGNMENT request. For more information on these APIs, see [Teradata Dynamic Workload Management APIs: PM/APIs](#).

**ProxyUser**

Name of the proxy user in a trusted session.

**CPUDecayLevel**

Current most severe decay level as reached due to CPU usage.

Nodes can be at different levels of decay (for example, 0, 1, or 2).

**IODecayLevel**

Current most severe decay level as reached due to I/O usage.

Nodes can be at different levels of decay (for example, 0, 1, or 2).

**TacticalCPUException**

Number of nodes that encountered a CPU exception.

**TacticalIOException**

Number of nodes that encountered an I/O exception.

**ReqIOKB**

Total logical I/O usage in KB.

**ReqPhysIO**

Number of physical I/Os.

**ReqPhysIOKB**

Physical I/O usage in KB.

**ReqStepsCompletedCnt**

Count of completed steps for the current request. No change in ReqStepsCompletedCnt from the previous Monitor Session collection indicates no new steps completed.

**RedriveProtection**

Redrive protection type:

- '' = No Redrive protection. This indicates that the session will not participate in Redrive and database restarts will not be transparent to applications and users.
- MN = Memory-based Redrive protection, no fallback spools

### **CurrentRedriveParticipation**

Indicates if the session is participating in Redrive. Sessions that use Redrive can enable or disable the functionality using the REDRIVE reserved query band. Possible values:

- T = Redrive functionality is enabled (database restarts are transparent to applications and users)
- F = Redrive functionality is disabled (database restarts are not transparent to applications and users)

### **ReqRedriveSpoolSpace**

Persistent spool space for the current request.

### **BlockerSessionCnt**

Total number of blocker sessions for the session. Unlike the MONITOR SESSION request, this field returns only the first three blocker sessions in the record parcel. There are no additional blocker sessions if there are more than three blocker sessions.

### **ReqTblOpBytesIn**

The total number of bytes transferred into the database from a foreign server for the current request through one or more table operators.

The request may involve one or multiple table operator executions. The ReqTblOpBytesIn output parameter shows bytes transferred across all invocations within the request.

### **ReqTblOpBytesOut**

The total number of bytes transferred out of the database and into a foreign server for the current request through one or more table operators.

The request may involve one or multiple table operator executions. The ReqTblOpBytesOut output parameter shows bytes transferred across all invocations within the request.

### **ProxyUserId**

The user ID charged for SPOOL and TEMP space if being charged to the proxy user.

### **Zoneld**

The unique identifier of the zone.

**ReqHotAmpCPU**

The CPU time of the highest CPU utilized AMP during the life of the current request on the session.

**ReqHotAmpCPUId**

Vproc ID of the highest CPU utilized AMP for the current request.

**ReqHotAmpIO**

I/O count of the highest I/O utilized AMP during the life of the current request on the session.

**ReqHotAmpIOld**

Vproc ID of the highest I/O utilized AMP for the current request.

**ReqInvolvedAMPCnt**

The number of AMPs involved in processing the current request.

**ReqFirstRespTime**

Date and time that the first response of the current request on the session is ready. The response may be held to meet the TASM Minimum Response Time.

**ReqLocalQueryStatus**

The current state of the Unified Data Architecture (UDA) query.

**ReqRemoteHostId**

Host ID of the remote system.

**ReqRemoteSessionId**

Session ID of the executing remote query.

**ReqRemoteRequestId**

Request ID of the executing remote query.

**ReqRemoteQueryId**

Query ID of the executing remote query.

**ReqHotAmpSpool**

Spool value of the highest spool utilized AMP during the life of the current request on the session.

This value is NULL if no request is running on the session.



**ReqHotAmpSpoolId**

Vproc ID of the highest spool utilized AMP for the current request.

This value is NULL if no request is running on the session.

**ReqMapNo**

Map number for the largest map the request is using.

**ReqMaxNumMapAMPs**

Number of AMPs in the largest contiguous map used by the request.

**ReqMinNumMapAMPs**

Number of AMPs in the smallest contiguous map used by the request.

**ReqSysDefNumMapAMPs**

Number of AMPs in the system-default map used by the request.

**ReqRemoteHostIp**

Host IP address of the remote system.

**ReqServerName**

Name of the foreign server.

**ReqFlexReleased**

The TDWM Flex Throttle feature detects available system resources, overrides existing workload throttle limits and automatically releases queries from the delay queue. This minimizes the DBA manually managing the TASM delay queue. Note, only Workload throttles are overridden; all System level throttles are still honored.

Return values:

- 0: Indicates that the request was not released by TDWM Flex Throttles.
- 1: Indicates that the request was released by TDWM Flex Throttles.

**ReqAdmissionTime**

Returns the date and time when a request is admitted into the system by TDWM.

**Usage Notes**

The following CPU fields in the MonitorMySessions response are affected by the MonSesCPUNormalization field:

- AMPCPUSec

- AvgAmpCPUsec
- HotAmp1CPU
- HotAmp2CPU
- HotAmp3CPU
- LowAmp1CPU
- LowAmp2CPU
- LowAmp3CPU
- PECPUsec
- RequestAmpCPU

The MonSesCPUNormalization field, a DBS Control General Record field, controls whether normalized or non-normalized statistical CPU data is reported by the functions: MonitorMySessions and MonitorSession, and by the MONITOR SESSION request. For more information about the MonSesCPUNormalization field and CPU normalization, see the DBS Control utility in *Teradata Vantage™ - Database Utilities*, B035-1102.

For a complete description of these fields, see the topic that follows.

You can also refer to [MonitorSession](#) and [MONITOR SESSION](#) for a list of these CPU fields.

A Trusted Session enables a middle-tier application to switch the user on an already active database session to another user (proxy user). Once the user is switched, all subsequent requests uses the privileges and session attributes of the proxy user. MonitorMySessions returns the logon user name and ID in the UserName and UserId fields and the proxy user name and ID in the ProxyUserName and ProxyUserId fields.

When connected as a permanent proxy user, MonitorMySessions returns all information on sessions directly logged on as the permanent proxy user. It does not return the current session or any other proxy user sessions for the permanent proxy user.

### Example: Using MonitorMySessions

```
select * from table (monitormysessions()) as t1;
```

```
*** Query completed. 2 rows found. 116 columns returned.
```

```
*** Total elapsed time was 1 second.
```

```

      HostId      1
      SessionNo   1006
      LogonPENO   30718
      RunVprocNo  30718
      PartName    DBC/SQL
      PEstate     DISPATCHING
      LogonTime   2016/04/29 11:26:46.00
      UserId      1028
      LSN         0
      UserName    JCK

```

```

UserAccount DBC
  PECPUsec 2.400000000000000E-002
  XActCount 1.000000000000000E 000
  ReqCount 2.000000000000000E 000
ReqCacheHits 0.000000000000000E 000
  AMPState ACTIVE
  AMPCPUsec 1.120500000000000E 003
  AMPIO 3.496050000000000E 005
  ReqSpool 4.39090831360000E 010
Blk1HostId 0
Blk1SessNo 0
Blk1UserId 0
  Blk1Lmode
  Blk10type
Blk10bjDBID 0
Blk10bjTID 0
Blk1Status
Blk2HostId 0
Blk2SessNo 0
Blk2UserId 0
  Blk2Lmode
  Blk20type
Blk20bjDBID 0
Blk20bjTID 0
Blk2Status
Blk3HostId 0
Blk3SessNo 0
Blk3UserId 0
  Blk3Lmode
  Blk30type
Blk30bjDBID 0
Blk30bjTID 0
Blk3Status
MoreBlockers
  LogonSource (TCP/IP) f1a3 153.64.183.39 MYSYSTEM 9964
  HotAmp1CPU 3.332000000000000E 001
  HotAmp2CPU 3.328800000000000E 001
  HotAmp3CPU 3.328800000000000E 001
HotAmp1CPUId 2
HotAmp2CPUId 1
HotAmp3CPUId 0
  HotAmp1IO 1.036200000000000E 004
  HotAmp2IO 1.034600000000000E 004
  HotAmp3IO 1.032700000000000E 004

```

```

HotAmp1IOId      2
HotAmp2IOId      3
HotAmp3IOId      1
LowAmp1CPU       3.32760000000000E 001
LowAmp2CPU       3.32880000000000E 001
LowAmp3CPU       3.32880000000000E 001
LowAmp1CPUId     3
LowAmp2CPUId     1
LowAmp3CPUId     0
LowAmp1IO        1.03160000000000E 004
LowAmp2IO        1.03270000000000E 004
LowAmp3IO        1.03460000000000E 004
LowAmp1IOId      0
LowAmp2IOId      1
LowAmp3IOId      3
AvgAmpCPUsec     3.32930000000000E 001
AvgAmpIOCnt      1.03377500000000E 004
AmpCount         4
TempSpaceUsg     0.00000000000000E 000
ReqStartTime     2016/04/29 11:26:46.00
ReqCPU           1.12050000000000E 003
ReqIO            3.49605000000000E 005
ReqNo            3
WlcId            0
DontReclassifyFlag 255
ProxyUser
CPUDecayLevel    0
IODecayLevel     0
TacticalCPUException 0
TacticalIOException 0
ReqIOKB          4.34172850000000E 007
ReqPhysIO        9.09660000000000E 004
ReqPhysIOKB      3.97004920000000E 007
ReqStepsCompletedCnt 9
RedriveProtection MN
CurrentRedriveParticipation F
ReqRedriveSpoolSpace 0.00000000000000E 000
BlockerSessionCnt 0
ReqTblOpBytesIn  0.00000000000000E 000
ReqTblOpBytesOut 0.00000000000000E 000
ProxyUserId      0
ZoneId           0
ReqHotAmpCPU     2.80292000000000E 002
ReqHotAmpCPUId   3

```

```

    ReqHotAmpIO 8.75320000000000E 004
    ReqHotAmpIOId 3
    ReqInvolvedAMPCnt 4
    ReqFirstRespTime 0000/00/00 00:00:00.00
    ReqLocalQueryStatus 0
    ReqRemoteHostId 0
    ReqRemoteSessionId 0
    ReqRemoteRequestId 0
    ReqRemoteQueryId 0.00000000000000E 000
    ReqHotAmpSpool 1.09939000320000E 010
    ReqHotAmpSpoolId 3
    ReqMapNo 0
    ReqMaxNumMapAMPs 0
    ReqMinNumMapAMPs 0
    ReqSysDefNumMapAMPs 0
    ReqRemoteHostIp
    ReqServerName
    ReqFlexReleased 0
    HostId 1
    SessionNo 1010
    LogonPENO 30719
    RunVprocNo 30719
    PartName DBC/SQL
    PEstimate DISPATCHING
    LogonTime 2016/04/29 11:45:40.00
    UserId 1028
    LSN 0
    UserName JCK
    UserAccount DBC
    PECPUsec 3.00000000000000E-001
    XActCount 1.00000000000000E 000
    ReqCount 3.00000000000000E 000
    ReqCacheHits 0.00000000000000E 000
    AMPState ACTIVE
    AMPCPUsec 4.00000000000000E-003
    AMPIO 0.00000000000000E 000
    ReqSpool 0.00000000000000E 000
    Blk1HostId 0
    Blk1SessNo 0
    Blk1UserId 0
    Blk1Lmode
    Blk1Otype
    Blk1ObjDBID 0
    Blk1ObjTID 0

```

```

Blk1Status
Blk2HostId      0
Blk2SessNo      0
Blk2UserId      0
Blk2Lmode
Blk2Otype
Blk2ObjDBID     0
Blk2ObjTID      0
Blk2Status
Blk3HostId      0
Blk3SessNo      0
Blk3UserId      0
Blk3Lmode
Blk3Otype
Blk3ObjDBID     0
Blk3ObjTID      0
Blk3Status
MoreBlockers
LogonSource (TCP/IP) f1f0 153.64.183.39 MYSYS      8348 JK
HotAmp1CPU      4.000000000000000E-003
HotAmp2CPU      0.000000000000000E 000
HotAmp3CPU      0.000000000000000E 000
HotAmp1CPUId    2
HotAmp2CPUId    0
HotAmp3CPUId    0
HotAmp1IO       0.000000000000000E 000
HotAmp2IO       0.000000000000000E 000
HotAmp3IO       0.000000000000000E 000
HotAmp1IOId     0
HotAmp2IOId     0
HotAmp3IOId     0
LowAmp1CPU      0.000000000000000E 000
LowAmp2CPU      0.000000000000000E 000
LowAmp3CPU      0.000000000000000E 000
LowAmp1CPUId    3
LowAmp2CPUId    1
LowAmp3CPUId    0
LowAmp1IO       0.000000000000000E 000
LowAmp2IO       0.000000000000000E 000
LowAmp3IO       0.000000000000000E 000
LowAmp1IOId     2
LowAmp2IOId     3
LowAmp3IOId     1
AvgAmpCPUSec    1.000000000000000E-003

```

```

    AvgAmpIOCnt  0.00000000000000E 000
    AmpCount      4
    TempSpaceUsg  0.00000000000000E 000
    ReqStartTime  2016/04/29 11:45:44.00
    ReqCPU        4.00000000000000E-003
    ReqIO         0.00000000000000E 000
    ReqNo         3
    WlcId         0
    DontReclassifyFlag 255
    ProxyUser
    CPUDecayLevel 0
    IODecayLevel  0
    TacticalCPUException 0
    TacticalIOException 0
    ReqIOKB       0.00000000000000E 000
    ReqPhysIO     0.00000000000000E 000
    ReqPhysIOKB   0.00000000000000E 000
    ReqStepsCompletedCnt 0
    RedriveProtection MN
CurrentRedriveParticipation F
    ReqRedriveSpoolSpace 0.00000000000000E 000
    BlockerSessionCnt    0
    ReqTblOpBytesIn      0.00000000000000E 000
    ReqTblOpBytesOut     0.00000000000000E 000
    ProxyUserId          0
    ZoneId               0
    ReqHotAmpCPU         4.00000000000000E-003
    ReqHotAmpCPUId       2
    ReqHotAmpIO          0.00000000000000E 000
    ReqHotAmpIOId        0
    ReqInvolvedAMPCnt    4
    ReqFirstRespTime     0000/00/00 00:00:00.00
    ReqLocalQueryStatus  0
    ReqRemoteHostId      0
    ReqRemoteSessionId   0
    ReqRemoteRequestId    0
    ReqRemoteQueryId     0.00000000000000E 000
    ReqHotAmpSpool       0.00000000000000E 000
    ReqHotAmpSpoolId     0
    ReqMapNo             0
    ReqMaxNumMapAMPs     0
    ReqMinNumMapAMPs     0
    ReqSysDefNumMapAMPs  0
    ReqRemoteHostIp

```

|                 |   |
|-----------------|---|
| ReqServerName   |   |
| ReqFlexReleased | 0 |

## MonitorPhysicalConfig

Collects overall information on node availability. Node status information is returned for all nodes in the system.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorPhysicalConfig (
) RETURNS TABLE (
  ProcId INTEGER,
  Status CHAR CHARACTER SET LATIN,
  CPUPType VARCHAR(7) CHARACTER SET LATIN,
  CPUCount SMALLINT,
  SystemType VARCHAR(7) CHARACTER SET LATIN,
  CliqueNo SMALLINT,
  NetAUp CHAR(1) CHARACTER SET LATIN,
  NetBUp CHAR(1) CHARACTER SET LATIN,
  PhyMemory INTEGER
)
...
;

```

### Syntax Elements

#### ProcId

ID associated with a node.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

#### Status

Status of the node associated with this record:

- U = Up/online
- D = Down/offline
- S = Standby

A node is up (U) when it is:

- Configured into the system
- Online
- Capable of actively performing tasks associated with normal database activity



Down (D) represents all other potential states.

Standby (S) indicates the node is ready to join the configuration in place if another node goes down. When the node status is Standby, the SystemType, NetAUp, and NetBUp fields are not available and NULL or spaces will be returned.

### CPUType

Type of CPU installed in this node, for example: 'Pentium', 'PentPro', or 'Unknown'.

### CPUCount

Number of CPUs in this node.

### SystemType

Type of system running the database software, such as 5650, 6700, or 'Other'.

If all the nodes in the system are the same type, this field returns the type of the system.

If any of the nodes are of a different type, this field returns 'Mixed'.

### CliqueNo

Clique number of the node.

### NetAUp

### NetBUp

Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:

- U = All node BYNETs are up/online.
- D = One or more node BYNETs is down/offline.
- "" = A temporary condition where the BYNET data is not available.

### PhyMemory

Size of the physical memory of the node in MBs.

## Usage Notes

The MonitorPhysicalConfig function provides similar functionality to the PMPC MONITOR PHYSICAL CONFIG request. For information about this interface, see [MONITOR PHYSICAL CONFIG](#).

### Example: Using MonitorPhysicalConfig

```
SELECT t2.* FROM TABLE (MonitorPhysicalConfig()) AS t2;
```

```
*** Query completed. One row found. 9 columns returned.
*** Total elapsed time was 1 second.
```

```

    ProcId    10001
    Status    U
    CPUType   Xeon
    CPUCount   1
    SystemType 5500C
    CliqueNo   0
    NetAUp
    NetBUp
    PhyMemory 5720

```

## MonitorPhysicalResource

Collects RSS data and returns node-specific data.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorPhysicalResource (
) RETURNS TABLE (
    ProcId INTEGER,
    Status CHAR CHARACTER SET LATIN,
    AmpCount SMALLINT,
    PECOUNT SMALLINT,
    CPUUse FLOAT,
    PrcntKernel FLOAT,
    PrcntService FLOAT,
    PrcntUser FLOAT,
    DiskUse FLOAT,
    DiskReads FLOAT,
    DiskWrites FLOAT,
    DiskOutReqAvg FLOAT,
    NetAUse FLOAT,
    NetReads FLOAT,
    NetWrites FLOAT,
    HstBlkRd FLOAT,
    HstBlkWrts FLOAT,
    MemAllocates FLOAT,
    MemAllocateKB FLOAT,
    MemFailures FLOAT,
    MemAgings FLOAT,
    NetAUp CHAR(1) CHARACTER SET LATIN,
    NetBUp CHAR(1) CHARACTER SET LATIN
)

```

```
...
;
```

## Syntax Elements

### Procid

ID associated with a node.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

### Status

Status of the node associated with this record:

- U = Up/online
- D = Down/offline

A node is up (U) when it is:

- Configured into the system
- Online
- Capable of actively performing tasks associated with normal database activity

Down (D) represents all other potential states.

### AmpCount

Current number of AMPs currently executing on the associated node.

### PECount

Current number of active PEs on the associated node.

### CPUUse

% of CPU usage not spent being idle. The node-level display is computed from ResUsageSpma table data as PercntUser + PercntService.

This value is NULL if certain conditions apply, see usage notes.

### PrcntKernel

% of CPU resources in idle and waiting for I/O completion. This value is computed from ResUsageSpma data as follows, where *NCPUs* is the number of CPUs:

$$((\text{CPU} \text{ IOWAIT} / 100) / (\text{NCPUs} * \text{SampleSec})) * 100$$

This value is NULL if certain conditions apply, see usage notes.

**PrcntService**

% of CPU resources spent in PDE user service processing. The value is computed from the ResUsageSpma table data, where  $x$  represents the number of CPUs:

$$\text{CPUUServ} / (x * \text{SampleSec})$$

This value is NULL if certain conditions apply, see usage notes.

**PrcntUser**

% of CPU resources spent in non-service user code processing. This value is computed from the ResUsageSpma table data, where  $x$  represents the number of CPUs:

$$\text{CPUUExec} / (x * \text{SampleSec})$$

This value is NULL if certain conditions apply, see usage notes.

**DiskUse**

% of disk usage per node.

This value is computed from ResUsageSldv table data as follows, assuming  $n$  is the number of vdisks used by this AMP:

$$(\text{LdvOutReqTime } 1 + \dots + \text{LdvOutReqTime } n) / (n * \text{SampleSec})$$

The DiskUse field does not take into account overlapping of operations among multiple storage devices, but it allows for the possibility of multiple requests for the same device.

This value is NULL if certain conditions apply, see usage notes.

**DiskReads**

Total number of physical disk reads per node during the collection period. This value is computed from ResUsageSldv table data as follows, assuming  $n$  is the number of ldv devices used by this node:

$$\text{LdvReads } 1 + \dots + \text{LdvReads } n$$

This value is NULL if certain conditions apply, see usage notes.

**DiskWrites**

Total number of physical disk writes per node during the collection period. This value is computed from ResUsageSldv table data as:

$$\text{LdvWrites } 1 + \dots + \text{LdvWrites } n$$

This value is NULL if certain conditions apply, see usage notes.

**DiskOutReqAvg**

Average number of outstanding disk requests.

The value is computed from ResUsageSldv table data as follows, assuming  $n$  is the number of Ldv controllers used by this node:

$$((\text{LdvOutReqSum } 1 / \text{NULLIFZERO}(\text{LdvOutReqDiv } 1)) + \dots + (\text{LdvOutReqSum } n / \text{NULLIFZERO}(\text{LdvOutReqDiv } n))) / n$$

The range of the value is typically 0 to 25.

This value is NULL if certain conditions apply, see usage notes.

### NetAUse

% of BYNET A usage. This is the actual BYNET receiver usage. (The BYNET transmitter usage is maintained in resource usage separately and is typically lower than the receiver usage. This is caused by multicasts, where one transmitter sends a message to many receivers.) This value is computed from the ResUsageSpma table data as:

$$((\text{NetSamples} - \text{NetTxIdle}) / \text{NetSamples}) * 100$$

This value is NULL if certain conditions apply, see usage notes.

### NetReads

Number of Reads from the BYNET to the node. This value is computed from the ResUsageSpma table data as:

$$\text{NetRxCircBrd} + \text{NetRxCircPtP}$$

This value is NULL if certain conditions apply, see usage notes.

### NetWrites

Number of messages written from the node to the BYNET during the collection period.

The value is computed from the ResUsageSpma table data as  $\text{NetTxCircBrd} + \text{NetTxCircPtP}$ .

This value is NULL if certain conditions apply, see usage notes.

### HstBlkRds

Number of message blocks (one or more messages sent in one physical group) received from all clients.

This value is computed from ResUsageShst data, assuming  $n$  is the number of host channel and network connections on this node:

$$\text{HostBlockReads } _1 + \dots + \text{HostBlockReads } _n$$

This value is NULL if certain conditions apply, see usage notes.

**HstBlkWrts**

Number of message blocks (that is, one or more messages sent in one physical group) sent to all hosts.

This value is computed from ResUsageShst data, assuming  $n$  is the number of host channel and network connections on this node:

$\text{HostBlockWrites}_1 + \dots + \text{HostBlockWrites}_n$

This value is NULL if certain conditions apply, see usage notes.

**MemAllocates**

This column is obsolete and returns zero or NULL.

**MemAllocateKB**

Value represents the change in node-level memory. MemAllocateKB represents a delta from the previous reporting period. It reports negative values as less memory is used.

This value is calculated from the ResUsageSpma column MemVprAllocKB.

This value is NULL if certain conditions apply, see usage notes.

**MemFailures**

This column is obsolete and returns zero or NULL.

**MemAgings**

This column is obsolete and returns zero or NULL.

**NetAUp****NetBUp**

Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:

- U = All node BYNETs are up/online.
- D = One or more node BYNETs is down/offline.
- "" = A temporary condition where the BYNET data is not available.

**Usage Notes**

The MonitorPhysicalResource function returns the detailed resource usage information for each node. Because the function reports on each node, you can isolate performance concerns by node.

The MonitorPhysicalResource function provides similar functionality to the PMPC MONITOR PHYSICAL RESOURCE request. For information about this interface, see [MONITOR PHYSICAL RESOURCE](#).

**Example: Using MonitorPhysicalResource**

```
SELECT t2.* from table (MonitorPhysicalResource()) as t2;
```

```
*** Query completed. One row found. 28 columns returned.
```

```
*** Total elapsed time was 1 second.
```

|                |                       |
|----------------|-----------------------|
| ProcId         | 10001                 |
| Status         | U                     |
| AmpCount       | 4                     |
| PECount        | 2                     |
| CPUUse         | 1.00000000000000E 002 |
| PrcntKernel    | 0.00000000000000E 000 |
| PrcntService   | 2.34804329725229E 000 |
| PrcntUser      | 9.76519567027477E 001 |
| DiskUse        | 9.08909242298085E 000 |
| DiskReads      | 8.48000000000000E 002 |
| DiskWrites     | 6.56500000000000E 003 |
| DiskOutReqAvg  | 1.58467943380516E-001 |
| NetAUse        | 0.00000000000000E 000 |
| NetBUse        | 0.00000000000000E 000 |
| NetReads       | 0.00000000000000E 000 |
| NetWrites      | 0.00000000000000E 000 |
| CICUse         | 0.00000000000000E 000 |
| HstBlkRds      | 2.00000000000000E 000 |
| HstBlkWrts     | 1.00000000000000E 000 |
| MemAllocates   | 0.00000000000000E 000 |
| MemAllocateKB  | 1.28000000000000E 002 |
| MemFailures    | 0.00000000000000E 000 |
| MemAgings      | 0.00000000000000E 000 |
| NVMemAllocate  | 0.00000000000000E 000 |
| NVMemAllocSegs | 0.00000000000000E 000 |
| NVMemAgings    | 0.00000000000000E 000 |
| NetAUp         | U                     |
| NetBUp         | U                     |

**MonitorPhysicalSummary**

Collects global summary information that includes the following types of information:

- CPU usage (average, high, and low)
- Disk usage (average, high, and low)
- BYNET usage (total, up/down)

- Rate information (resource logging rate and resource monitoring rate)
- Current software release and version numbers

## Syntax

```

REPLACE FUNCTION SYSLIB.MonitorPhysicalSummary (
) RETURNS TABLE (
    AvgCPU FLOAT,
    AvgDisk FLOAT,
    AvgDiskIO FLOAT,
    HighCPUUse FLOAT,
    HighDisk FLOAT,
    HighDiskIO FLOAT,
    HighCPUProcId INTEGER,
    HighDiskProcId INTEGER,
    HighDiskIOProcId INTEGER,
    LowCPUUse FLOAT,
    LowDisk FLOAT,
    LowDiskIO FLOAT,
    LowCPUProcId INTEGER,
    LowDiskProcId INTEGER,
    LowDiskIOProcId INTEGER,
    NetUse FLOAT,
    NetAUp CHAR(1) CHARACTER SET LATIN,
    NetBUp CHAR(1) CHARACTER SET LATIN,
    ResLogging SMALLINT,
    ResMonitor SMALLINT,
    ReleaseNum VARCHAR(30) CHARACTER SET LATIN,
    Version VARCHAR(32) CHARACTER SET LATIN
)
...
;

```

## Syntax Elements

### AvgCPU

Average % CPU usage (CPUUse) time of all online nodes currently in the database configuration.

This value is NULL if certain conditions apply, see usage notes.

### AvgDisk

Average % disk usage (from DiskUse) of all online nodes currently in the database configuration.



Assuming  $n$  is the number of online AMPs in the configuration, AMPAvgDisk is computed from DiskUse data as:

$$(\text{DiskUse}_1 + \dots + \text{DiskUse}_n) / n$$

This value is NULL if certain conditions apply, see usage notes.

### AvgDiskIO

Average physical disk DiskReads and DiskWrites of all online AMPs in the configuration.

Assuming  $n$  is the number of online AMPs in the configuration, AMPAvgDiskIO is computed from DiskReads and DiskWrites data as:

$$(\text{DiskReads}_1 + \text{DiskWrites}_1 + \dots + \text{DiskReads}_n + \text{DiskWrites}_n) / n$$

This value is NULL if certain conditions apply, see usage notes.

### HighCPUUse

Highest CPUUse number associated with any online node that is currently part of the database configuration.

This value is NULL if certain conditions apply, see usage notes.

### HighDisk

Highest % disk usage (from DiskUse) associated with any online node that is currently part of the database configuration.

This value is NULL if certain conditions apply, see usage notes.

### HighDiskIO

ID of a node with DiskReads and DiskWrites equal to the value reported as HighDiskIO.

This value is NULL if certain conditions apply, see usage notes.

### HighCPUProclD

ID of a node with CPPUse equal to the value reported as HighCPUUse.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

### HighDiskProclD

ID of a node with DiskUse equal to the value reported as HighDisk.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**HighDiskIOProcId**

ID of a node with DiskReads and DiskWrites equal to the value reported as HighDiskIO.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**LowCPUUse**

Lowest CPUUse number associated with any online node that is currently part of the database configuration.

This value is NULL if certain conditions apply, see usage notes.

**LowDisk**

Lowest % disk usage (from DiskUse) associated with any online node that is currently part of the database configuration.

This value is NULL if certain conditions apply, see usage notes.

**LowDiskIO**

Lowest DiskReads and DiskWrites number associated with any online node that is currently part of the database configuration.

This value is NULL if certain conditions apply, see usage notes.

**LowCPUProcId**

ID of a node with CPUUse equal to the value reported as LowCPUUse.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**LowDiskProcId**

ID of a node with DiskUse equal to the value reported as LowDisk.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**LowDiskIOProocId**

ID of a node with DiskReads and DiskWrites equal to the value reported as LowDiskIO.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**NetUse**

% of total BYNET use (that is, average of the online BYNETs).

If both BYNETs are up, the value is computed from ResUsageSpma table data as:

$\text{NetUse} = \text{Average NetAUse per node} / \text{NetCount}$

where:

- *NetCount* is 2 if both NetA and NetB are up or 1 if only one of the BYNET is up.
- *Average NetAUse* is the sum of all NetAUse of each node divided by the number of online nodes.

This value is NULL if certain conditions apply, see usage notes.

NetUse returns a value of zero because resource usage data is not currently available.

**NetAUp****NetBUp**

Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:

- U = All node BYNETs are up/online.
- D = One or more node BYNETs is down/offline.
- "" = A temporary condition where the BYNET data is not available.

**ResLogging**

Interval in seconds at which resource usage data is written to one or more active resource usage database tables.

**ResMonitor**

Interval in seconds at which all resource usage data is collected in memory for reporting via the PM/API.

**ReleaseNum**

Release number of the currently running database software (for example, 15.00.00.00).

This value is supplied by the database.

**Version**

Version number of the currently running database software (for example, 15.00.00.00).

This value is supplied by the database.

**Usage Notes**

The MonitorPhysicalSummary function provides similar functionality to the PMPC MONITOR PHYSICAL SUMMARY request. For information about this interface, see [MONITOR PHYSICAL SUMMARY](#).

**Example: Using MonitorPhysicalSummary**

```
SELECT * FROM TABLE (MonitorPhysicalSummary()) AS t1;
```

```
*** Query completed. One row found. 22 columns returned.
```

```
*** Total elapsed time was 1 second.
```

```

      AvgCPU    1.00000000000000E 002
      AvgDisk   7.51986754966887E 000
      AvgDiskIO 6.64700000000000E 003
      HighCPUUse 1.00000000000000E 002
      HighDisk   7.51986754966887E 000
      HighDiskIO 6.64700000000000E 003
      HighCPUTProcId      10001
      HighDiskProcId      10001
      HighDiskIOProcId    10001
      LowCPUUse   1.00000000000000E 002
      LowDisk     7.51986754966887E 000
      LowDiskIO   6.64700000000000E 003
      LowCPUTProcId      10001
      LowDiskProcId      10001
      LowDiskIOProcId    10001
      NetUse      0.00000000000000E 000
      NetAUp      U
      NetBUp      U
      ResLogging   60
      ResMonitor   60
      ReleaseNum   16t.00.00.97
      Version     16t.00.00.97_dr182707k

```

**MonitorSession**

Returns session or request resource usage statistics.

**Syntax**

```

REPLACE FUNCTION SYSLIB.MonitorSession (
  HostIdIn SMALLINT,
  UserNameIn TD_ANYTYPE,
  SessionNoIn INTEGER
) RETURNS TABLE (
  HostId SMALLINT,
  SessionNo INTEGER,
  LogonPENO SMALLINT,
  RunVprocNo SMALLINT,
  PartName VARCHAR(16) CHARACTER SET LATIN,
  PEstate VARCHAR(18) CHARACTER SET LATIN,
  LogonTime VARCHAR(22) CHARACTER SET LATIN,
  UserId INTEGER,
  LSN INTEGER,
  UserName VARCHAR(128) CHARACTER SET UNICODE,
  UserAccount VARCHAR(128) CHARACTER SET UNICODE,
  PECPUsec FLOAT,
  XActCount FLOAT,
  ReqCount FLOAT,
  ReqCacheHits FLOAT,
  AMPState VARCHAR(18) CHARACTER SET LATIN,
  AMPCPUsec FLOAT,
  AMPPIO FLOAT,
  ReqSpool FLOAT,
  Blk1HostId SMALLINT,
  Blk1SessNo INTEGER,
  Blk1UserId INTEGER,
  Blk1Lmode CHAR(1) CHARACTER SET LATIN,
  Blk1Otype CHAR(2) CHARACTER SET LATIN,
  Blk1ObjDBID INTEGER,
  Blk1ObjTID INTEGER,
  Blk1Status CHAR(1) CHARACTER SET LATIN,
  Blk2HostId SMALLINT,
  Blk2SessNo INTEGER,
  Blk2UserId INTEGER,
  Blk2Lmode CHAR(1) CHARACTER SET LATIN,
  Blk2Otype CHAR(2) CHARACTER SET LATIN,
  Blk2ObjDBID INTEGER,
  Blk2ObjTID INTEGER,
  Blk2Status CHAR(1) CHARACTER SET LATIN,
  Blk3HostId SMALLINT,

```

```

Blk3SessNo INTEGER,
Blk3UserId INTEGER,
Blk3Lmode CHAR(1) CHARACTER SET LATIN,
Blk3Otype CHAR(2) CHARACTER SET LATIN,
Blk3ObjDBID INTEGER,
Blk3ObjTID INTEGER,
Blk3Status CHAR(1) CHARACTER SET LATIN,
MoreBlockers CHAR(1) CHARACTER SET LATIN,
LogonSource VARCHAR(128),
HotAmp1CPU FLOAT,
HotAmp2CPU FLOAT,
HotAmp3CPU FLOAT,
HotAmp1CPUId FLOAT,
HotAmp2CPUId FLOAT,
HotAmp3CPUId FLOAT,
HotAmp1IO FLOAT,
HotAmp2IO FLOAT,
HotAmp3IO FLOAT,
HotAmp1IOId INTEGER,
HotAmp2IOId INTEGER,
HotAmp3IOId INTEGER,
LowAmp1CPU FLOAT,
LowAmp2CPU FLOAT,
LowAmp3CPU FLOAT,
LowAmp1CPUId INTEGER,
LowAmp2CPUId INTEGER,
LowAmp3CPUId INTEGER,
LowAmp1IO FLOAT,
LowAmp2IO FLOAT,
LowAmp3IO FLOAT,
LowAmp1IOId INTEGER,
LowAmp2IOId INTEGER,
LowAmp3IOId INTEGER,
AvgAmpCPUSec FLOAT,
AvgAmpIOCnt FLOAT,
AmpCount SMALLINT,
TempSpaceUsg FLOAT,
ReqStartTime VARCHAR(22) CHARACTER SET LATIN,
ReqCPU FLOAT,
ReqIO FLOAT,
ReqNo INTEGER,
WlcId INTEGER,
DontReclassifyFlag SMALLINT,
ProxyUser VARCHAR (128) CHARACTER SET UNICODE,

```

```

CPUDecayLevel SMALLINT,
IODecayLevel SMALLINT,
TacticalCPUException INTEGER,
TacticalIOException INTEGER,
ReqIOKB FLOAT,
ReqPhysIO FLOAT
ReqPhysIOKB FLOAT
ReqStepsCompletedCnt INTEGER,
RedriveProtection CHAR (2) CHARACTER SET LATIN,
CurrentRedriveParticipation CHAR (1) CHARACTER SET LATIN,
ReqRedriveSpoolSpace FLOAT,
BlockerSessionCnt SMALLINT,
ReqTblOpBytesIn FLOAT,
ReqTblOpBytesOut FLOAT,
ProxyUserId INTEGER,
ZoneId INTEGER,
ReqHotAmpCPU FLOAT,
ReqHotAmpCPUId SMALLINT,
ReqHotAmpIO FLOAT,
ReqHotAmpIOId SMALLINT,
ReqInvolvedAMPCnt SMALLINT,
ReqFirstRespTime VARCHAR(22) CHARACTER SET LATIN,
ReqLocalQueryStatus SMALLINT,
ReqRemoteHostId SMALLINT,
ReqRemoteSessionId INTEGER,
ReqRemoteRequestId INTEGER,
ReqRemoteQueryId FLOAT)
ReqHotAmpSpool FLOAT,
ReqHotAmpSpoolId SMALLINT,
ReqMapNo SMALLINT,
ReqMaxNumMapAMPs INTEGER,
ReqMinNumMapAMPs INTEGER,
ReqSysDefNumMapAMPs INTEGER,
ReqRemoteHostIp VARCHAR(128) CHARACTER SET UNICODE,
ReqServerName VARCHAR(128) CHARACTER SET UNICODE,
ReqFlexReleased SMALLINT,
ReqAdmissionTime VARCHAR(22) CHARACTER SET LATIN
)
...
;

```

## Syntax Elements

### ***HostIdIn***

Logical ID of a host (or client) with sessions logged on.

A value of -1 indicates all hosts.

### ***UserNameIn***

User name of the session.

An asterisk (\*) indicates all users.

### ***SessionNoIn***

Number of the session.

A value of zero indicates all sessions.

### **HostId**

Logical host ID associated with a PE or session. For a PE, HostId identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs. A value of zero represents the Supervisor window.

### **SessionNo**

Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.

### **LogonPENo**

Vproc number of the PE the session is currently logged on to; it identifies the PE that has control responsibility for the session. Normally, this is the PE that processed the logon request; but if that PE goes offline, it will be the PE to which the session was switched.

### **RunVprocNo**

Vproc number of the AMP or PE currently assigned to process the session requests.

For sessions in Teradata SQL partitions, this value is identical to the LogonPENo. For sessions in FastLoad or MultiLoad partitions, this is the AMP that initially processes the data being loaded.



If a RunVprocNo value of -1 in record mode or NULL in indicator mode is returned by MonitorSession for FastLoad, MultiLoad or FastExport sessions, this indicates that the session is in the process of starting up.

### PartName

Name of the session partition associated with this session. Following a successful logon request by a session or as part of a connect request, the session identifies the partition with which the user wants the session to be associated. FASTLOAD, Teradata SQL, MONITOR, INTERNAL are examples of valid partition names.

### PEState

Current state of the session within the PE. See the MONITOR SESSION PESTate field for a list of the PESTate values.

### LogonTime

Date and time portion of the information recorded by Session Control when a session successfully logs on. It is usually formatted for display as *yyyy/mm/dd 99:99:99.99*, which represents the year, month, day, hours: minutes: seconds.

### UserId

User or internal ID of a user for this session. Within the database, UserId is equivalent to Database ID. Typically, UserId is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, UserId can identify either a given user or database name.

### LSN

Logon Sequence Number (LSN) that is associated with a session when the session logs on. It identifies a collection of sessions performing a related activity. For example, in a FastLoad job, a user is logged on as a Teradata SQL session, as well as  $n$  FastLoad sessions with the same user name. Therefore,  $n+1$  sessions (1 Teradata SQL and  $n$  FastLoad) with the same LSN are all associated with the given FastLoad job. To see how the FastLoad job is doing, the user can pick out all sessions reported with the same LSN number.

This information supplies the parent-child relationship for sessions involved with FastLoad and MultiLoad jobs.

### UserName

User name of the session.

### UserAccount

Current account for the session.

**PECPUsec**

CPU time, in seconds, used in a PE by the associated session for parsing and dispatching requests. It is accurate to the second.

This value is valid only when associated with Teradata SQL and MONITOR partition sessions.

**XActCount**

Number of explicit and implicit transactions executed by the session.

This value is valid only when returned for Teradata SQL sessions, and is NULL for all other partition sessions. For this value, you must make a request for data before completion of the first collection period that follows either a system outage or a change in the ResMonitor rate.

**ReqCount**

Number of requests (Tequel Start Request (TSR) messages) initiated by the session.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or change in the ResMonitor rate.

**ReqCacheHits**

Number of times that this session processed a request using information from the Teradata SQL Parser request cache, specifically, how many times there was a request cache hit.

This value is valid only for Teradata SQL sessions, and is NULL for all other partition sessions. This value is also NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

**AMPState**

Current state of the associated session in AMP vprocs in decreasing priority:

| State    | Description                                                                                   |
|----------|-----------------------------------------------------------------------------------------------|
| ABORTING | Transaction being rolled back; session aborting.                                              |
| BLOCKED  | Background activity in progress; last request on hold until background activity is completed. |
| ACTIVE   | Normal, on-going activity.                                                                    |
| IDLE     | No work in progress on any AMP.                                                               |
| UNKNOWN  | No recorded activity since monitoring began.                                                  |

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

### **AMPCPUSec**

Current elapsed CPU time, in seconds, used on all AMPs by the associated session for executing requests. For example, for Teradata SQL requests, this is the time spent by the database actively working or rolling back an aborted transaction. This does not include any PDE CPU time spent handling database requests.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

### **AMPIO**

Current number of logical Reads and Writes issued across all AMPs by the associated session.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

### **ReqSpool**

Current spool used by current request across all AMPs, expressed as a number of bytes.

This value is NULL when a request for data is made before completion of the first collection period following either a system outage or a change in the ResMonitor rate.

### **Blk1HostId**

### **Blk2HostId**

### **Blk3HostId**

Logical host ID of a session causing a block. This value is derived from equating the transactions causing a database lock conflict to the sessions that issued those transactions. The Blk\_x\_HostId in combination with Blk\_x\_SessNo uniquely identifies the session that is causing a block.

This value is NULL if:

- The host ID is not available.
- The session does not have an AMPState of BLOCKED.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserId values all return as NULLs and AMPState is BLOCKED, a Host Utility (HUT) lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1SessNo****Blk2SessNo****Blk3SessNo**

Number of the session causing a block. This value is derived from associating the transactions causing a lock conflict to the sessions that issued those transactions. The Blk\_x\_SessNo in combination with Blk\_x\_HostId uniquely identifies the session causing a block.

This value is NULL if:

- SessNo is not available.
- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1UserId****Blk2UserId****Blk3UserId**

ID of the user or host utility job preventing the session from being granted a lock. The user ID is the only information available about who placed the lock.

This value is NULL if:

- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

If the Blk\_x\_HostId, Blk\_x\_SessNo, and Blk\_x\_UserID values all return as NULLs and AMPState is BLOCKED, a host utility lock left over after the session holding the lock was aborted or logged off. The lock was never released, and no blocking information is available because the session no longer exists.

Use the Show Locks utility to obtain the user name that placed the HUT lock. For more information, see *Teradata Vantage™ - Database Utilities*, B035-1102.

**Blk1Lmode****Blk2Lmode****Blk3Lmode**

Mode (severity) of the lock involved in causing a block:

- E = Exclusive
- W = Write
- R = Read
- A = Access

Locks are reported in decreasing order of severity because removing the most severe lock conflict may eliminate the source of the lock conflict.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

A session may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock in the queue and is in conflict with the lock requested by this blocked session. For information on whether the lock is granted, see the MONITOR SESSION fields: Blk1Status, Blk2Status, and Blk3Status.

**Blk1Otype****Blk2Otype****Blk3Otype**

Type of object whose lock is causing the session described by the associated row to be blocked:

- D = Database
- T= Table
- R = Row hash
- TP = Table Partition range
- RP = RowHash in Partition range
- RK = RowHash in one partition
- RN = RowKey range

However, this object is not necessarily the type of object the blocked session is trying to access. For example, if the session is requesting a row hash lock, the blocking object could be a database, table, or row hash.

This value is NULL if:

- Session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

For a Table T, it is possible for User A to block User B with a table level lock on Table T on AMP\_1 and with a Row Hash Level lock on that same Table T on AMP\_2. When that occurs, the only lock conflict reported is that User B is blocked by User A on a table.

**Blk1ObjDBID****Blk2ObjDBID****Blk3ObjDBID**

Unique ID of the database object over which a lock conflict is preventing the session from being granted a lock.

Within the database system, Database ID is equivalent to User ID. Typically, User ID is used when the associated record is known to be a user name, and Database ID is used when the associated record is known to be a database. However, Database ID can identify either a user or database name.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**Blk1ObjTID****Blk2ObjTID****Blk3ObjTID**

Unique ID of the table object over which a lock conflict is preventing the session from being granted a lock.

This value is NULL if:

- The session does not have an AMPState of BLOCKED.
- The Blk\_x\_OType is D.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

**Blk1Status****Blk2Status****Blk3Status**

Status of lock causing a block:

- W= Waiting
- G = Granted

This value is NULL if:

- Session does not have an AMPState of BLOCKED.

- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

A lock request may be blocked by either a granted lock or an ungranted lock that precedes the blocked lock request in the queue and is in conflict with it.

A status of Waiting has a higher priority than that of Granted when there is more than one lock involved. For example, for a given object and a given session, a session that is blocked by a Waiting lock on one AMP and a Granted lock on another AMP has Waiting reported as its status.

### MoreBlockers

Indicator of more lock conflicts:

- Blank = Blk x information is a complete list of sessions blocking the session described.
- Asterisk (\*) = Additional sessions are blocking the session described.

This value is NULL if:

- The state of the session is not BLOCKED.
- A request for data is made before completion of the first collection period following either a system outage or a change in the SesMonitorLoc or SesMonitorSys rate.

### LogonSource

Logon source information. At logon time, this information is optionally supplied by the Teradata Director Program or the Gateway to further identify the physical or logical location of the session, the logon user name, and the interface under which the session was initiated. (For example, the data string may include DBC as the user ID and BTEQ as the interface.)

A two-byte value precedes the LogonSource data to indicate the length of the string. The length value is zero if LogonSource is NULL.

For a list of the commonly seen LogonSource string application names, see *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

### HotAmp $n$ CPU

Where  $n$  in  $[1, 3]$ : CPU time of the  $n$ th highest CPU utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### HotAmp $n$ CPUId

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th highest CPU utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### HotAmpnIO

Where  $n$  in  $[1, 3]$ : I/O count of the  $n$ th highest I/O utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### HotAmpnIOld

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th highest I/O utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### LowAmpnCPU

Where  $n$  in  $[1, 3]$ : CPU time of the  $n$ th lowest CPU utilized AMP during the collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### LowAmpnCPUld

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th lowest CPU utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### LowAmpnIO

Where  $n$  in  $[1, 3]$ : I/O count of the  $n$ th lowest I/O utilized AMP during the collection interval.



Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **LowAmpnIOld**

Where  $n$  in  $[1, 3]$ : Vproc ID of the  $n$ th lowest I/O utilized AMP for the last session collection interval.

Where  $n$  in  $[1, 2]$ : This value is NULL if the request is made before the collection period expires.

Where  $n = 3$ : This value is NULL if the request is made before the collection period expires, and if there are only two AMPs on the system.

### **AvgAmpCPUsec**

Average AMP CPU utilization for the last session collection interval. The average is calculated as the sum of CPU utilization for all amps participating divided by the number of online AMPs.

This value is NULL if the request is made before the collection period expires.

### **AvgAmpIOCnt**

Average AMP I/O utilization for the last session collection interval. The average is calculated as the sum of I/O utilization for all AMPs participating divided by the number of online AMPs.

This value is NULL if the request is made before the collection period expires.

### **AmpCount**

Current number of AMPs currently executing on the associated node.

### **TempSpaceUsg**

Total amount, in bytes, of temporary space used by the session.

This value is NULL if the session did not materialize any temporary tables.

### **ReqStartTime**

Date and time of the current request on the session started. It is usually formatted for display as `yyyy/mm/dd 99:99:99.99`, which represents the year, month, day hours: minutes: seconds.

### **ReqCPU**

Total CPU usage by the current SQL request on the session on all AMPs. This value contains proper request-level statistics for DBC/SQL sessions running SQL requests only. Ignore the

value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.

This field is equivalent to the MONITOR SESSION RequestAmpCPU field.

### **ReqIO**

Total number of accesses by the current SQL request for the session on all AMPs. This value contains proper request-level statistics for DBC/SQL sessions running SQL requests only. Ignore the value returned in this field for other types of sessions, such as DBC/SQL sessions linked to a utility job.

This field is equivalent to the MONITOR SESSION RequestAmpI/O field.

### **ReqNo**

Active request number.

If no request is running, a value of zero or NULL is displayed in indicator mode.

In some rare cases, in the very early phase of a request in parsing state, when PESTate = PARSING, the request number may not be available and returned as zero or NULL. The active request number will be available on the next collection.

### **WlId**

Workload ID associated with the specified request.

### **DontReclassifyFlag**

Flag indicating that the next request on the session will not be classified but will use the workload ID (WlId) already assigned to the session. This will occur if this is a utility session or a WlId was assigned to the session using the TDWMAssignWD function or the TDWM WD ASSIGNMENT request. For more information on these APIs, see [Teradata Dynamic Workload Management APIs: PM/APIs](#).

### **ProxyUser**

Name of the proxy user in a trusted session.

### **CPUDecayLevel**

Current most severe decay level as reached due to CPU usage.

Nodes can be at different levels of decay (for example, 0, 1, or 2).

### **IODecayLevel**

Current most severe decay level as reached due to I/O usage.

Nodes can be at different levels of decay (for example, 0, 1, or 2).

**TacticalCPUException**

Number of nodes that encountered a CPU exception.

**TacticalIOException**

Number of nodes that encountered an I/O exception.

**ReqIOKB**

Total logical I/O usage in KB.

**ReqPhysIO**

Number of physical I/Os.

**ReqPhysIOKB**

Physical I/O usage in KB.

**ReqStepsCompletedCnt**

Count of completed steps for the current request. If there is no change in ReqStepsCompletedCnt from the previous Monitor Session collection, this indicates that there are no new steps completed.

**RedriveProtection**

Redrive protection type:

- '' = No Redrive protection. This indicates that the session will not participate in Redrive and database restarts will not be transparent to applications and users.
- MN = Memory-based Redrive protection, no fallback spools

**CurrentRedriveParticipation**

Indicates if the session is participating in Redrive. Sessions that use Redrive can enable or disable the functionality using the REDRIVE reserved query band. Possible values:

- T = Redrive functionality is enabled (database restarts are transparent to applications and users)
- F = Redrive functionality is disabled (database restarts are not transparent to applications and users)

**ReqRedriveSpoolSpace**

Persistent spool space for the current request.

**BlockerSessionCnt**

Total number of blocker sessions for the session. Unlike the MONITOR SESSION request, this field returns only the first three blocker sessions in the record parcel. There are no additional blocker sessions if there are more than three blocker sessions.

**ReqTblOpBytesIn**

The total number of bytes transferred into the database from a foreign server for the current request through one or more table operators.

The request may involve one or multiple table operator executions. The ReqTblOpBytesIn output parameter shows bytes transferred across all invocations within the request.

**ReqTblOpBytesOut**

The total number of bytes transferred out of the database and into a foreign server for the current request through one or more table operators.

The request may involve one or multiple table operator executions. The ReqTblOpBytesOut output parameter shows bytes transferred across all invocations within the request.

**ProxyUserId**

The UserID charged for SPOOL and TEMP space if being charged to the proxy user.

**Zoneld**

The unique identifier of the zone.

**ReqHotAmpCPU**

The CPU time of the highest CPU utilized AMP during the life of the current request on the session.

**ReqHotAmpCPUID**

Vproc ID of the highest CPU utilized AMP for the current request.

**ReqHotAmpIO**

I/O count of the highest I/O utilized AMP during the life of the current request on the session.

**ReqHotAmpIOID**

Vproc ID of the highest I/O utilized AMP for the current request.

**ReqInvolvedAMPCnt**

The number of AMPs involved in processing the current request.

**ReqFirstRespTime**

Date and time that the first response of the current request on the session is ready. The response may be held to meet the TASM Minimum Response Time.

**ReqLocalQueryStatus**

The current state of the Unified Data Architecture (UDA) query.

**ReqRemoteHostId**

Host ID of the remote system.

**ReqRemoteSessionId**

Session ID of the executing remote query.

**ReqRemoteRequestId**

Request ID of the executing remote query.

**ReqRemoteQueryId**

Query ID of the executing remote query.

**ReqHotAmpSpool**

Spool value of the highest spool utilized AMP during the life of the current request on the session.

This value is NULL if no request is running on the session.

**ReqHotAmpSpoolId**

Vproc ID of the highest spool utilized AMP for the current request.

This value is NULL if no request is running on the session.

**ReqMapNo**

Map number for the largest map the request is using.

**ReqMaxNumMapAMPs**

Number of AMPs in the largest contiguous map used by the request.

**ReqMinNumMapAMPs**

Number of AMPs in the smallest contiguous map used by the request.

**ReqSysDefNumMapAMPs**

Number of AMPs in the system-default map used by the request.

**ReqRemoteHostIp**

Host IP address of the remote system.

**ReqServerName**

Name of the foreign server.

**ReqFlexReleased**

The TDWM Flex Throttle feature detects available system resources, overrides existing workload throttle limits and automatically releases queries from the delay queue. This minimizes the DBA manually managing the TASM delay queue. Note, only Workload throttles are overridden; all System level throttles are still honored.

Return values:

- 0: Indicates that the request was not released by TDWM Flex Throttles.
- 1: Indicates that the request was released by TDWM Flex Throttles.

**ReqAdmissionTime**

Returns the date and time when a request is admitted into the system by TDWM.

**Usage Notes**

This table function is only supported in Constant Mode.

The following CPU fields in the MonitorSession response are affected by the MonSesCPUNormalization field:

- AMPCPUSec
- AvgAmpCPUSec
- HotAmp1CPU
- HotAmp2CPU
- HotAmp3CPU
- LowAmp1CPU
- LowAmp2CPU
- LowAmp3CPU
- PECPUSec
- RequestAmpCPU

The MonSesCPUNormalization field, a DBS Control General Record field, controls whether normalized or non-normalized statistical CPU data is reported by the functions: MonitorSession and MonitorMySessions, and by the MONITOR SESSION request. For more information about the MonSesCPUNormalization field and CPU normalization, see the DBS Control utility in *Teradata Vantage™ - Database Utilities*, B035-1102.

For a complete description of these fields, see the topic that follows.

You can also refer to [MonitorMySessions](#) or [MONITOR SESSION](#) for a list of these CPU fields.

A Trusted Session enables a middle-tier application to switch the user on an already active database session to another user (proxy user). Once the user is switched, all subsequent requests use the privileges and session attributes of the proxy user. MonitorSession returns the logon user name and ID in the UserName and UserId fields and the proxy user name and ID in the ProxyUserName and ProxyUserId fields.

The MonitorSession function provides similar functionality to the PMPC MONITOR SESSION request. For information about this interface, see [MONITOR SESSION](#).

### Example: Using MonitorSession

```
sel * from table (monitorsession(-1,'*',0)) as t1 where t1.username='jck';
```

```
*** Query completed. One row found. 116 columns returned.
```

```
*** Total elapsed time was 1 second.
```

```

      HostId      1
      SessionNo   1006
      LogonPENO   30718
      RunVprocNo  30718
      PartName    DBC/SQL
      PEstate     DISPATCHING
      LogonTime   2016/04/29 11:26:46.00
      UserId      1028
      LSN         0
      UserName    JCK
      UserAccount DBC
      PECPUsec    2.400000000000000E-002
      XActCount   1.000000000000000E 000
      ReqCount    2.000000000000000E 000
      ReqCacheHits 0.000000000000000E 000
      AMPState    ACTIVE
      AMPCPUsec   9.873280000000000E 002
      AMPPIO      3.082540000000000E 005
      ReqSpool    3.870573260800000E 010
      Blk1HostId  0
      Blk1SessNo  0
      Blk1UserId  0
      Blk1Lmode
      Blk1Otype
      Blk1ObjDBID 0
      Blk1ObjTID  0
      Blk1Status
      Blk2HostId  0
      Blk2SessNo  0

```

```

Blk2UserId          0
Blk2Lmode
Blk2Otype
Blk2ObjDBID         0
Blk2ObjTID          0
Blk2Status
Blk3HostId          0
Blk3SessNo          0
Blk3UserId          0
Blk3Lmode
Blk3Otype
Blk3ObjDBID         0
Blk3ObjTID          0
Blk3Status
MoreBlockers
LogonSource (TCP/IP) f1a3 153.64.183.39 MYSYSTEM      9964
HotAmp1CPU 1.568400000000000E 001
HotAmp2CPU 1.567600000000000E 001
HotAmp3CPU 1.567600000000000E 001
HotAmp1CPUId      3
HotAmp2CPUId      1
HotAmp3CPUId      2
HotAmp1IO 4.899000000000000E 003
HotAmp2IO 4.897000000000000E 003
HotAmp3IO 4.889000000000000E 003
HotAmp1IOId       2
HotAmp2IOId       3
HotAmp3IOId       0
LowAmp1CPU 1.567600000000000E 001
LowAmp2CPU 1.567600000000000E 001
LowAmp3CPU 1.567600000000000E 001
LowAmp1CPUId      1
LowAmp2CPUId      2
LowAmp3CPUId      0
LowAmp1IO 4.884000000000000E 003
LowAmp2IO 4.889000000000000E 003
LowAmp3IO 4.897000000000000E 003
LowAmp1IOId       1
LowAmp2IOId       0
LowAmp3IOId       3
AvgAmpCPUSec 1.567800000000000E 001
AvgAmpIOCnt 4.892250000000000E 003
AmpCount          4
TempSpaceUsg 0.000000000000000E 000

```



```

ReqStartTime 2016/04/29 11:26:46.00
ReqCPU 9.873280000000000E 002
ReqIO 3.082540000000000E 005
ReqNo 3
WlcId 0
DontReclassifyFlag 255
ProxyUser
CPUDecayLevel 0
IODecayLevel 0
TacticalCPUException 0
TacticalIOException 0
ReqIOKB 3.827829300000000E 007
ReqPhysIO 8.014500000000000E 004
ReqPhysIOKB 3.500234900000000E 007
ReqStepsCompletedCnt 9
RedriveProtection MN
CurrentRedriveParticipation F
ReqRedriveSpoolSpace 0.000000000000000E 000
BlockerSessionCnt 0
ReqTblOpBytesIn 0.000000000000000E 000
ReqTblOpBytesOut 0.000000000000000E 000
ProxyUserId 0
ZoneId 0
ReqHotAmpCPU 2.470160000000000E 002
ReqHotAmpCPUId 3
ReqHotAmpIO 7.718600000000000E 004
ReqHotAmpIOId 3
ReqInvolvedAMPCnt 4
ReqFirstRespTime 0000/00/00 00:00:00.00
ReqLocalQueryStatus 0
ReqRemoteHostId 0
ReqRemoteSessionId 0
ReqRemoteRequestId 0
ReqRemoteQueryId 0.000000000000000E 000
ReqHotAmpSpool 9.691859456000000E 009
ReqHotAmpSpoolId 3
ReqMapNo 0
ReqMaxNumMapAMPs 0
ReqMinNumMapAMPs 0
ReqSysDefNumMapAMPs 0
ReqRemoteHostIp
ReqServerName
ReqFlexReleased 0

```

## MonitorSessionRate

Returns session rate (duration of the collection period in seconds).

### Syntax

```
REPLACE FUNCTION SYSLIB.MonitorSessionRate (
  HostIdIn SMALLINT,
  UserNameIn TD_ANYTYPE,
  SessionNoIn INTEGER
) RETURNS SMALLINT
  ...
;
```

### Syntax Elements

#### *HostIdIn*

Logical ID of a host (or client) with sessions logged on.

A value of -1 indicates all hosts.

#### *UserNameIn*

User name of the specified sessions.

A value of -1 indicates all hosts.

#### *SessionNoIn*

Number of the specified session.

A value of zero indicates all sessions.

### Usage Notes

Before you monitor the collection rate, you use the SetSessionRate function to set the collection rate for updating session-level statistics in memory.

The MonitorSessionRate function provides similar functionality to the PMPC Monitor Session request. For information about this interface, see [MONITOR SESSION](#).

### Example: Using MonitorSessionRate

```
SELECT MonitorSessionRate(1, '*', 0);
*** Query completed. One row found. One column returned.
*** Total elapsed time was 7 seconds.
MonitorSessionRate(1, '*', 0)
```

-----  
60

## MonitorSQLCurrentStep

Returns data about the step being executed of the currently running request for the specified host, session, and vproc.

### Syntax

```
REPLACE FUNCTION SYSLIB.MonitorSQLCurrentStep (
    HostIdIn SMALLINT,
    SessionNoIn INTEGER,
    RunVprocNo SMALLINT
) RETURNS TABLE (
    HostId SMALLINT,
    SessionNo INTEGER,
    DynamicPlan SMALLINT
    PartialSteps SMALLINT,
    NumOfSteps SMALLINT,
    CurLvl1StepNo SMALLINT,
    CurLvl2StepNo SMALLINT
    ZoneID INTEGER,
    SPName VARCHAR(128) CHARACTER SET UNICODE,
    SPDBName VARCHAR(128) CHARACTER SET UNICODE
    DefaultDBName VARCHAR(128) CHARACTER SET UNICODE
)
...
;
```

### Syntax Elements

#### *HostIdIn*

Logical ID of a host (or client) with sessions logged on.

A value of -1 indicates all hosts.

#### *SessionNoIn*

Session number of the SQL to monitor.

#### *RunVprocNo*

PE vproc number where the session runs.

**HostId**

Logical host ID associated with a PE or session. For a PE, HostId identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs. A value of zero represents the Supervisor window.

**SessionNo**

Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.

**DynamicPlan**

Plan type:

- 0 = Static plan
- 1 = Dynamic plan

For more information on static and dynamic explanations of a request, see the EXPLAIN request modifier in *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146 or *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142.

**PartialSteps**

Possible values:

- 0 = All steps are returned
- 1 = Partial plan or no plan is returned

If a partial plan is returned, this indicates the steps for the final plan fragment of the dynamic explanation of the request has not yet been generated.

If no plan is returned, this indicates the request has been throttled and is in the delay queue.

A value 1 cannot occur for a static plan.

For more information on static and dynamic explanations of a request, see *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146 or *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142.

**NumOfSteps**

Number of steps in the description text in the third statement of the response.

| Plan   | DynamicPlan<br>Field Value | PartialSteps<br>Field Value | NumOfSteps                |
|--------|----------------------------|-----------------------------|---------------------------|
| Static | 0                          |                             | Number of steps for plan. |

| Plan                | DynamicPlan<br>Field Value | PartialSteps<br>Field Value | NumOfSteps                 |
|---------------------|----------------------------|-----------------------------|----------------------------|
| Complete<br>dynamic | 1                          | 0                           | Number of steps for plan.  |
| Partial dynamic     | 1                          | 1                           | Number of steps generated. |

If a request with a partial dynamic plan has been throttled and is in the delay queue (that is, no rows are returned in response to the third statement), NumOfSteps is zero.

For more information, see the MONITOR SQL DynamicPlan and PartialSteps fields.

### CurLvl1StepNo

Number of the currently executing level 1 step. If parallel steps are executing, it is the number of the lowest executing step.

If this is a request with a dynamic plan that has been throttled and is in the delay queue (for example, when the NumOfSteps field value is zero and both the DynamicPlan are PartialSteps field values are 1), the CurLvl1StepNo field value is zero.

### CurLvl2StepNo

Number of the currently executing step. If parallel steps are executing, it is the number of the highest executing step. If only one step is executing, CurLvl1StepNo and CurLvl2StepNo are identical.

If this is a request with a dynamic plan that has been throttled and is in the delay queue (for example, when the NumOfSteps field value is zero and both the DynamicPlan are PartialSteps field values are 1), the CurLvl2StepNo field value is 1.

---

#### Note:

If only one step is executing, CurLvl1StepNo and CurLvl2StepNo are identical.

---

### Zoneld

The unique identifier of the zone.

### SPName

The outer stored procedure name, if a stored procedure is being executed.

NULL is returned in indicator mode if no stored procedure is being executed.

### SPDName

This is the owner database name of the outer stored procedure if a stored procedure is being executed.

NULL is returned in indicator mode if no stored procedure is being executed.

### DefaultDBName

This field returns the default database name of the session at the start of the non-stored procedure request. For stored procedures, it returns the default database name of the session when the stored procedure was compiled.

### Usage Notes

This table function is only supported in Constant Mode.

If MONITOR SQL processing is not completed within the timeout interval, then an error is returned to the client application. When a MONITOR SQL request is timed out, the processing continues internally to its completion. If the client application submits a new MONITOR SQL request for the same timed out target session while the previous timed out one is still being processed, then an error is returned. The timeout interval can be set in the DBS Control field, PMPC\_TimeoutSecs. The default timeout interval is 60 seconds. If the PMPC\_TimeoutSecs field is set to zero, the MONITOR SQL timeout request will be disabled and no timeout will occur. For more information on the PMPC\_TimeoutSecs field, see *Teradata Vantage™ - Database Utilities*, B035-1102.

The MonitorSQLCurrentStep function provides similar functionality to the PMPC MONITOR SQL request. For information about this interface, see [MONITOR SQL](#).

### Example: Using MonitorSQLCurrentStep

```
select * from table (monitorsqlcurrentstep(1,1472,30718)) as t1;
```

```
*** Query completed. One row found. 11 columns returned.
```

```
*** Total elapsed time was 1 second.
```

|               |          |
|---------------|----------|
| HostId        | 1        |
| SessionNo     | 1472     |
| DynamicPlan   | 0        |
| PartialSteps  | 0        |
| NumOfSteps    | 13       |
| CurLvl1StepNo | 10       |
| CurLvl2StepNo | 10       |
| ZoneId        | 0        |
| SPName        |          |
| SPDBName      |          |
| DefaultDBName | TESTUSER |

## MonitorSQLSteps

Returns the step information (that is, a scaled-down version of the output of the EXPLAIN request modifier) of the current or running request for the specified host, session, and vproc.

### Syntax

```
REPLACE FUNCTION SYSLIB.MonitorSQLSteps (
  HostIdIn SMALLINT,
  SessionNoIn INTEGER,
  RunVprocNo SMALLINT
) RETURNS TABLE (
  HostId SMALLINT,
  SessionNo INTEGER,
  DynamicPlan SMALLINT,
  PartialSteps SMALLINT
  StepNum INTEGER,
  Confidence SMALLINT,
  EstRowCount FLOAT,
  ActRowCount FLOAT,
  EstRowCountSkew FLOAT,
  ActRowCountSkew FLOAT,
  EstRowCountSkewMatch FLOAT,
  ActRowCountSkewMatch FLOAT,
  EstElapsedTime FLOAT,
  ActElapsedTime FLOAT,
  SQLStep VARCHAR(2048) CHARACTER SET UNICODE
)
...
;
```

### Syntax Elements

#### *HostIdIn*

Logical ID of a host (or client) with sessions logged on.

#### *SessionNoIn*

Session number of the SQL to monitor.

#### *RunVprocNo*

PE vproc number where the session runs.

**HostId**

Logical host ID associated with a PE or session. For a PE, HostId identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs. A value of zero represents the Supervisor window.

**SessionNo**

Number of the current session. Together with a given host ID, a session number uniquely identifies a session on the database system. This value is assigned by the host (or client) at logon time.

**DynamicPlan**

Plan type:

- 0 = Static plan
- 1 = Dynamic plan

For more information on static and dynamic explanations of a request, see the EXPLAIN request modifier in *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146 or *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142.

**PartialSteps**

Possible values:

- 0 = All steps are returned
- 1 = Partial plan or no plan is returned

If a partial plan is returned, this indicates the steps for the final plan fragment of the dynamic explanation of the request has not yet been generated.

If no plan is returned, this indicates the request has been throttled and is in the delay queue.

A value 1 cannot occur for a static plan.

For more information on static and dynamic explanations of a request, see *Teradata Vantage™ - SQL Data Manipulation Language*, B035-1146 or *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142.

**StepNum**

Unique number identifying the EXPLAIN step.

**Confidence**

Confidence level as determined by the optimizer:

- 0 = None



- 1= Foreign Key
- 2 = Low
- 3 = High

**EstRowCount**

Estimated row count generated from the Optimizer plan for this step.

For a PRPD plan, the EstRowCount field for the split step (that is, a RETRIEVE or JOIN step with “split into” appearing in the EXPLAIN when target spools are generated) is the estimated row counts for ALL split spools.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

**ActRowCount**

Actual row count returned from the AMP for this step.

For a PRPD plan, it includes rows from all split spools for a split step.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

**EstRowCountSkew**

Estimated row count for the skew split spool in PRPD, which contains the rows with skewed values of this spool.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

**ActRowCountSkew**

Actual number of rows for the skew split spool in PRPD.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

**EstRowCountSkewMatch**

Estimated row count for the skew match split spool in PRPD, which contains the rows with skewed values of the other relation to be joined with this relation.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

**ActRowCountSkewMatch**

Actual number of rows for the skew match split spool in PRPD.

For more information on PRPD, see *Teradata Vantage™ - SQL Request and Transaction Processing*, B035-1142 or *Teradata Vantage™ - Database Administration*, B035-1093.

### EstElapsedTime

Estimated time for the query as generated from the Optimizer plan.

### ActElapsedTime

Actual elapsed time calculated by the dispatcher.

### SQLStep

Generated text for the step.

## Usage Notes

This table function is only supported in Constant Mode.

If MONITOR SQL processing is not completed within the timeout interval, then an error is returned to the client application. When a MONITOR SQL request is timed out, the processing continues internally to its completion. If the client application submits a new MONITOR SQL request for the same timed out target session while the previous timed out one is still being processed, then an error is returned. The timeout interval can be set in the DBS Control field, PMPC\_TimeoutSecs. The default timeout interval is 60 seconds. If the PMPC\_TimeoutSecs field is set to zero, the MONITOR SQL timeout request will be disabled and no timeout will occur. For more information on the PMPC\_TimeoutSecs field, see *Teradata Vantage™ - Database Utilities*, B035-1102.

The MonitorSQLSteps function provides similar functionality to the PMPC MONITOR SQL request. For information about this interface, see [MONITOR SQL](#).

## Example: Using MonitorSQLSteps

```
select StepNum (format '99') Num,
Confidence (format '9') C,
EstRowCount (format '-99999999') ERC,
ActRowCount (format '99999999') ARC,
EstRowCountSkew (format '-99999999') ERCS,
ActRowCountSkew (format '99999999') ARCS,
EstRowCountSkewMatch (format '-99999999') ERCSM,
ActRowCountSkewMatch (format '99999999') ARCSM,
EstElapsedTime (format '99999') EET,
ActElapsedTime (format '99999') AET,
SQLStep
from table (MonitorSQLSteps(1, 1164, 30719)) as t2;
*** Query completed. 33 rows found. 11 columns returned.
*** Total elapsed time was 3 seconds.
```

| Num | C | ERC      | ARC      | ERCS     | ARCS     | ERCSM    | ARCSM    | EET    | AET    | SQLStep                                                  |
|-----|---|----------|----------|----------|----------|----------|----------|--------|--------|----------------------------------------------------------|
| 01  | 0 | 00000000 | 00000004 | 00000000 | 00000000 | 00000000 | 00000000 | 000000 | 000000 | First, lock [DBId=0x0407]. for exclusive.                |
| 02  | 0 | 00000000 | 00000001 | 00000000 | 00000001 | 00000000 | 00000001 | 000000 | 000000 | Next, we lock DBC.[TBId=0x0130] for write on a row hash. |
| 03  | 0 | 00000000 | 00000001 | 00000000 | 00000001 | 00000000 | 00000001 | 000000 | 000000 | We lock DBC.DBSpace for write on a row hash.             |

```

04 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We lock DBC
.Parents for write on a row hash.
05 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We lock DBC
.Owners for write on a row hash.
06 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We lock DBC
.AccessRights for write on a row hash.
07 0 00000000 00000004 00000000 00000000 00000000 00000000 00000 00000 We lock DBC.
[TBId=0x0130] for write, we lock DBC.DBSpace for write, we lock DBC.Parents for write,
we lock DBC.Owners for write, we lock DBC.DBase for write on a row hash, we lock DBC.DBase for write on a row ha
sh, we lock DB
08 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step begins a p
arallel block of steps.
08 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-AMP ABORT test from DBC.
[TBId=0x0138] by way of the unique primary index. This step is performed in parallel.
08 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step is perform
ed in parallel.
08 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP ABORT test from DBC.DBase by way of the unique primary index. This step is perform
ed in parallel.
08 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do an IN
SERT step into table DBC.DBase. This step is performed in parallel.
08 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP UPDATE from DBC.DBase by way of the unique primary index. This step is performed i
n parallel.
08 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP RETRIEVE step from DBC.Parents by way of the primary index into Spool 54, which is
redistributed by hash code to few AMPs. This step ends a parallel block of steps.
09 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We do a MER
GE into table DBC.Owners from Spool 54.
10 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do an IN
SERT step into table DBC.Owners. This step begins a parallel block of steps.
10 0 00000000 00000000 00000000 00000001 00000000 00000001 00000 00000 We do a Sin
gle-
AMP RETRIEVE step from DBC.Parents by way of the primary index into Spool 55, which is
redistributed by hash code to few AMPs. This step ends a parallel block of steps.
11 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We do a MER
GE into table DBC.Parents from Spool 55.
12 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do an IN
SERT step into table DBC.Parents. This step begins a parallel block of steps.
12 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do an IN
SERT step into table DBC.Accounts. This step is performed in parallel.
12 0 00000000 00000025 00000000 00000000 00000000 00000000 00000 00000 We do a Sin
gle-
AMP RETRIEVE step from DBC.AccessRights accessing a single partition by way of the pri
mary index into Spool 56
, which is redistributed by hash code to few AMPs. This step ends a parallel block of
steps.
13 0 00000000 00000021 00000000 00000000 00000000 00000000 00000 00000 We do a Sin
gle-
AMP RETRIEVE step from DBC.AccessRights accessing a single partition by way of the pri
mary index into Spool 56
, which is redistributed by hash code to few AMPs. This step begins a parallel block o
f steps.
13 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We do an All-
AMPs RETRIEVE step from DBC.AccessRights by way of an all-
rows scan into Spool 57, which is redistributed by has
h code to all AMPs. This step ends a parallel block of steps.
14 0 00000000 00000046 00000000 00000000 00000000 00000000 00000 00000 We do an All-

```

```

AMPs JOIN step from DBC.Owners by way of an all-
rows scan, which is joined to Spool 57. table Owners and Spool 5
7 are joined using a merge join . The result goes into Spool 56, which is redistribute
d by hash code
15 0 00000000 00000046 00000000 00000000 00000000 00000000 00000 00000 We do a MER
GE into table DBC.AccessRights from Spool 56. This step begins a parallel block of ste
ps.
15 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We do an IN
SERT step into table [TBId=0x0130]. This step ends a parallel block of steps.
16 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We flush th
e DISKSPACE and AMPUSAGE caches.
17 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We do an All-
AMPs ABORT test from DBC.DBSpace by way of the unique primary index.
18 0 00000000 00000004 00000000 00000000 00000000 00000000 00000 00000 We do an IN
SERT step into table DBC.DBSpace.
19 0 00000000 00000004 00000000 00000000 00000000 00000000 00000 00000 We do an All-
AMPs UPDATE from DBC.DBSpace by way of the unique primary index.
20 0 00000000 00000000 00000000 00000000 00000000 00000000 00000 00000 We flush th
e DISKSPACE and AMPUSAGE caches.
21 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00001 We Spoil th
e parser's dictionary cache for the database.
22 0 00000000 00000001 00000000 00000001 00000000 00000001 00000 00000 We send out
an END TRANSACTION step to all AMPs involved in processing the request.

```

## MonitorSQLText

Returns the SQL text of the request currently being executed for the specified host, session, and vproc.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorSQLText (
  HostIdIn SMALLINT,
  SessionNoIn INTEGER,
  RunVprocNo SMALLINT
) RETURNS TABLE (
  HostId SMALLINT,
  SessionNo INTEGER,
  SeqNum SMALLINT,
  SQLTxt VARCHAR(31000) CHARACTER SET UNICODE
)
...
;

```

### Syntax Elements

#### *HostIdIn*

Logical ID of a host (or client) with sessions logged on.

#### *SessionNoIn*

Session number of the SQL to monitor.

**RunVprocNo**

PE vproc number where the session runs.

**HostId**

Host ID of the SQLtext.

**SessionNo**

Session number of the SQLtext.

**SeqNum**

Sequence number of the row. For example, if the SQL text exceeds 31,000 bytes, the system will return multiple rows.

**SQLTxt**

SQL text of the running request.

**Usage Notes**

This table function is only supported in Constant Mode.

If MONITOR SQL processing is not completed within the timeout interval, then an error is returned to the client application. When a MONITOR SQL request is timed out, the processing continues internally to its completion. If the client application submits a new MONITOR SQL request for the same timed out target session while the previous timed out one is still being processed, then an error is returned. The timeout interval can be set in the DBS Control field, PMPC\_TimeoutSecs. The default timeout interval is 60 seconds. If the PMPC\_TimeoutSecs field is set to zero, the MONITOR SQL timeout request will be disabled and no timeout will occur. For more information on the PMPC\_TimeoutSecs field, see *Teradata Vantage™ - Database Utilities*, B035-1102.

The MonitorSQLText function provides similar functionality to the PMPC MONITOR SQL request. For information about this interface, see [MONITOR SQL](#).

**Example: Using MonitorSQLText**

```
SELECT SQLTxt FROM TABLE (MonitorSQLText(1, 1001, 16383)) AS t2;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 58 seconds.
SQLTxt
-----
select a11.c1, a12.c1, a13.c1, a21.c1, a22.c1, a23.c1 from dbaaa.skewAmp1
a11, dbaaa.skewAmp1 a12, dbaaa.skewAmp1 a13, dbaaa.skewAmp1 a21, dbaaa.sk
ewAmp1 a22, dbaaa.skewAmp1 a23;
```

## MonitorSystemPhysicalConfig

Returns BYNET status, system type and name values that are generated once for the whole system.

### Syntax

```
REPLACE FUNCTION SYSLIB.MonitorSystemPhysicalConfig (
) RETURNS TABLE (
  NetAUp CHAR CHARACTER SET LATIN,
  NetBUp CHAR CHARACTER SET LATIN,
  SystemType VARCHAR(7) CHARACTER SET LATIN,
  SystemName VARCHAR(15) CHARACTER SET LATIN
)
...
;
```

### Syntax Elements

#### NetAUp

#### NetBUp

Status of the BYNETs (if there are more than two, the first two) on a system-wide basis:

- U = All node BYNETs are up/online.
- D = One or more node BYNETs is down/offline.
- "" = A temporary condition where the BYNET data is not available.

#### SystemType

Type of system running the database software, such as 5650, 6700, or 'Other'.

If all the nodes in the system are the same type, this field returns the type of the system.

If any of the nodes are of a different type, this field returns 'Mixed'.

#### SystemName

Name of the system running Teradata.

### Usage Notes

The MonitorSystemPhysicalConfig function provides similar functionality to statement 1 of the MONITOR PHYSICAL CONFIG request. For more information, see [MONITOR PHYSICAL CONFIG](#).

**Example: Using MonitorSystemPhysicalConfig**

```
SELECT * FROM table (SYSLIB.MonitorSystemPhysicalConfig()) as t1;
*** Query completed. One row found. 4 columns returned.
*** Total elapsed time was 1 second.
```

| NetAUp | NetBUp | SystemType | SystemName |
|--------|--------|------------|------------|
| U      | U      | 5500C      | localhost  |

**Example: Using MonitorSystemPhysicalConfig and Selecting a Specific Column**

```
SELECT SystemName FROM table (SYSLIB.MonitorSystemPhysicalConfig()) as t1;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
```

| SystemName |
|------------|
| localhost  |

**MonitorVirtualConfig**

Collects information on virtual processor (vproc) availability.

**Syntax**

```
REPLACE FUNCTION SYSLIB.MonitorVirtualConfig (
) RETURNS TABLE (
  ProcId INTEGER,
  VprocNo SMALLINT,
  Vproctype VARCHAR(3) CHARACTER SET LATIN,
  HostId SMALLINT,
  Status CHAR CHARACTER SET LATIN,
  DiskSlice SMALLINT
)
...
;
```

**Syntax Elements****ProcId**

ID associated with a node.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

### VprocNo

ID of an AMP (that is, a set of disks and the associated tasks or processes that, in combination, make up the AMP), PE, or TVS vproc.

### Vproctype

Type of vproc:

- AMP
- PE
- MISC

### HostId

Logical host ID associated with a PE or session. For a PE, HostId identifies one of the hosts or LANs associated with the described PE. For a session, the combination of a host ID and a session number uniquely identifies a user session on the system.

This value is NULL for AMPs and TVS vprocs. A value of zero represents the Supervisor window.

### Status

Status of the vproc associated with this record. A vproc is considered up or down from the standpoint of whether the vproc is helping a query process SQL statements. For example, an AMP doing offline recovery is considered to be down because the AMP is not helping to process SQL statements. On the other hand, an up vproc is one that is online and fully up or is in online recovery.

The status of the vproc:

- U = The vproc is up/online.
- D = The vproc is down/offline.

### DiskSlice

Virtual disk ID defining the portion of a physical disk assigned to an AMP.

This value is NULL for TVS vprocs.

## Usage Notes

The MonitorVirtualConfig function provides similar functionality to the PMPC MONITOR VIRTUAL CONFIG request. For more information, see [MONITOR VIRTUAL CONFIG](#).



**Example: Using MonitorVirtualConfig**

```
SELECT t2.* FROM TABLE (MonitorVirtualConfig()) AS t2;
```

```
*** Query completed. 8 rows found. 6 columns returned.
```

```
*** Total elapsed time was 1 second.
```

| ProcId | VprocNo | Vproctype | HostId | Status | DiskSlice |
|--------|---------|-----------|--------|--------|-----------|
| -----  | -----   | -----     | -----  | -----  | -----     |
| 10001  | 0       | AMP       | 0      | U      | 0         |
| 10001  | 1       | AMP       | 0      | U      | 1         |
| 10001  | 2       | AMP       | 0      | U      | 2         |
| 10001  | 3       | AMP       | 0      | U      | 3         |
| 10001  | 28670   | TVS       | 0      | U      | 0         |
| 10001  | 28671   | TVS       | 0      | U      | 0         |
| 10001  | 30718   | PE        | 1      | U      | 0         |
| 10001  | 30719   | PE        | 1      | U      | 0         |

**MonitorVirtualResource**

Collects performance information for each AMP, PE, or TVS vproc.

**Syntax**

```
REPLACE FUNCTION SYSLIB.MonitorVirtualResource (
) RETURNS TABLE (
  VprocNo SMALLINT,
  VprocType VARCHAR(3) CHARACTER SET LATIN,
  Status CHAR CHARACTER SET LATIN,
  ProcId INTEGER,
  ClusterNo SMALLINT,
  SessLogCount SMALLINT,
  SessRunCount SMALLINT,
  CPUUse FLOAT,
  PctService FLOAT,
  PctAMPWT FLOAT,
  DiskUse FLOAT,
  DiskReads FLOAT,
  DiskWrites FLOAT,
  DiskOutReqAvg FLOAT,
  PctParser FLOAT,
  PctDispatcher FLOAT,
  HstBlkRds FLOAT,
```

```

    HstBlkWrts FLOAT,
    NetReads  FLOAT,
    NetWrites FLOAT,
    NVMemAllocSegs FLOAT
  )
  ...
;

```

## Syntax Elements

### VprocNo

ID of an AMP (that is, a set of disks and the associated tasks or processes that, in combination, make up the AMP), PE or TVS vproc.

### VprocType

Type of vproc:

- AMP
- PE
- MISC

### Status

Status of the node, AMP, PE, or TVS vproc associated with this record:

- U = Up/online.
- D = Down/offline.

A node is up (U) when it is:

- Configured into the system
- Online
- Capable of actively performing tasks associated with normal database activity

Down (D) represents all other potential states.

### Procid

ID associated with a node.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

### ClusterNo

Number that identifies the cluster to which this AMP is assigned.

ClusterNo is not applicable to TVS vproc and returns NULL.

**SessLogCount**

Number of current sessions logged to this PE. A logged on session is either a session whose logon request was delivered to this PE, or a session that was switched to this PE following its logon.

The SessLogCount field contains the SubPoolId if the vproc type is TVS.

SubpoolId identifies the subpool associated with the allocator vproc. A subpool defines a set of storage and allocator vprocs assigned to that storage.

This value is NULL if certain conditions apply, see usage notes.

**SessRunCount**

Number of current sessions whose Initiate Requests (TSR messages) are addressed to this vproc. For example:

- PEs have a SessRunCount that includes all the Teradata SQL and MONITOR sessions logged on to that PE.
- AMPs may have a nonzero SessRunCount, since AMPs receive TSR messages from FastLoad or MultiLoad logons.

This value is NULL if certain conditions apply, see usage notes.

This field is not applicable to TVS vprocs.

**CPUUse**

% of CPU usage not spent being idle.

This value is computed from ResUsageSvpr table data, where *NCPUs* is the number of CPUs in the node:

$$100.00 * (\text{CPUUExecPart00} + \text{CPUUExecPart01} + \dots + \text{CPUUExecPart47} + \text{CPUUServPart00} + \text{CPUUServPart01} + \dots + \text{CPUUServPart47}) / (\text{NCPUs} * \text{SampleSec} * 100)$$

This value is NULL if certain conditions apply, see usage notes.

**PctService**

% of CPU resources spent in PDE user service processing.

This value is computed from the ResUsageSvpr table data, where *x* represents the number of CPUs:

$$(\text{CPUUServPart00} + \text{CPUUServPart01} + \dots + \text{CPUUServPart47}) / (x * \text{SampleSec})$$

This value is NULL if certain conditions apply, see usage notes.

**PctAMPWT**

% of CPU resources used by either the AMP Worker Task (Partition 11) or by the TVS Task (Partition 31) depending on the type of vproc this record represents.

This value depends on the number of CPUs in the node but does not exceed 100%. It is computed from the ResUsageSvpr table data, where  $x$  represents the number of CPUs on a node:

- For AMP vprocs:  

$$(\text{CPUUExecPart11}) / (x * \text{SampleSec})$$
- For TVS vprocs:  

$$(\text{CPUUExecPart31}) / (x * \text{SampleSec})$$

This value is NULL if certain conditions apply, see usage notes.

**DiskUse**

% of disk usage per AMP. This value is computed from the ResUsageSvdsk table data:

$$\text{OutReqTime} / \text{SampleSec}$$

DiskUse does not take into account overlapping of operations among multiple storage controllers, but it allows for the possibility of multiple requests for the same controller.

This value is NULL if certain conditions apply, see usage notes.

**DiskReads**

Total number of physical disk reads per AMP during the collection period.

This value is computed from the ResUsageSvpr table data as:

$$\text{FilePCiAcqReads} + \text{FilePDdbAcqReads} + \text{FileSciAcqReads} + \text{FileSDbAcqReads} + \text{FileTjtAcqReads} + \text{FileAPtAcqReads}$$

This value is NULL if certain conditions apply, see usage notes.

This field is not applicable to TVS and PE vprocs.

**DiskWrites**

Total number of physical disk writes per AMP during the collection period.

For PE and AMP-level displays, this value is computed from ResUsageSvpr table data as:

$$\begin{aligned} &\text{FilePCiFWrites} + \text{FilePDdbFWrites} + \text{FileSciFWrites} + \text{FilesDbFWrites} + \text{FileTjtFWrites} \\ &+ \text{FileAPtFWrites} + \text{FilePCiDyaWrites} + \text{FilePDdbDyaWrites} + \text{FileSciDyaWrites} + \\ &\text{FilesDbDyaWrites} + \text{FileTjtDyaWrites} + \text{FileAptDyaWrites} \end{aligned}$$

For TVS vproc displays, this value is computed from ResUsageSvpr table data as:

$$\text{AllocatorMapIOsDone}$$

This value is NULL if certain conditions apply, see usage notes.

### DiskOutReqAvg

Average number of outstanding disk requests.

For AMP-level displays, this value is computed from ResUsageSvdsk table data, assuming  $n$  is the number of storage devices used by this vproc:

$(\text{ReadRespTot } 1 + \text{WriteRespTot } 1 + \dots + \text{ReadRespTot } n + \text{WriteRespTot } n) / \text{CentiSecs}$

This field is not applicable to PE vprocs.

For TVS vproc-level displays, this value is computed from ResUsageSvpr table data as:

$\text{AllocatorMapIOsStarted} - \text{AllocatorMapIOsDone}$

This value is NULL if certain conditions apply, see usage notes.

### PctParser

This field is obsolete and returns zero or NULL.

### PctDispatcher

% of CPU resources spent in PE Dispatcher processing.

This value depends on the number of CPUs in the node but does not exceed 100%. This value is computed from the ResUsageSvpr table data, where  $x$  represents the number of CPUs on a node:

$(\text{CPUUExecPart13} + \text{CPUUServPart13}) / (x * \text{SampleSec})$

The PercntParser CPU time is included in the PercntDispatcher value.

This value is NULL if certain conditions apply, see usage notes.

This field is not applicable to AMP and TVS vprocs.

### HstBlkRds

Number of message blocks (one or more messages sent in one physical group) received from all clients.

This value corresponds to the column totals in the ResUsageShst table supplying HostBlockReads for this vproc.

This value is NULL if certain conditions apply, see usage notes.

This field is not applicable to AMP and TVS vprocs.

### HstBlkWrts

Number of message blocks (that is, one or more messages sent in one physical group) sent to all hosts.

This value corresponds to the column totals in the HstBlkWrts column of the ResUsageShst table.

This value is NULL if certain conditions apply, see usage notes.

This field is not applicable to AMP and TVS vprocs.

### NetReads

Number of Reads from the BYNET to the vproc.

The value is computed from the ResUsageSvpr table data as:

NetBrdReads + NetPtPReads

This value is NULL if certain conditions apply, see usage notes.

### NetWrites

Number of messages written from the AMP, PE, or vproc to the BYNET during the collection period.

The value is computed from the ResUsageSvpr table data as:

NetBrdWrites + NetPtPWrites

This value is NULL if certain conditions apply, see usage notes.

### NVMemAllocSegs

Value is computed from ResUsageSvpr table data using the IoRespMax field. The IoRespMax is the maximum I/O response time in milliseconds. That is, the number of operations for each AMP vproc on that node.

This field is not applicable to AMP and TVS vprocs.

This value is NULL if certain conditions apply, see usage notes.

The NVMemAllocSegs field of the MonitorVirtualResource function corresponds to the MaxIOResp field of the MONITOR VIRTUAL RESOURCE request.

## Usage Notes

The MonitorVirtualResource function provides similar functionality to the PMPC MONITOR VIRTUAL RESOURCE request. For information about this interface, see [MONITOR VIRTUAL RESOURCE](#).

For information on the resource usage tables and columns described in the field descriptions below, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

### Example: Using MonitorVirtualResource

```
SELECT VprocNo, VprocType, Status, PProcId, ClusterNo FROM
TABLE (MonitorVirtualResource()) AS t2;
```

```
*** Query completed. 8 rows found. 5 columns returned.
*** Total elapsed time was 1 second.
```

| VprocNo | VprocType | Status | ProcId | ClusterNo |
|---------|-----------|--------|--------|-----------|
| -----   | -----     | -----  | -----  | -----     |
| 0       | AMP       | U      | 10001  | 0         |
| 1       | AMP       | U      | 10001  | 1         |
| 2       | AMP       | U      | 10001  | 0         |
| 3       | AMP       | U      | 10001  | 1         |
| 28670   | TVS       | U      | 10001  | 0         |
| 28671   | TVS       | U      | 10001  | 0         |
| 30718   | PE        | U      | 10001  | 1025      |
| 30719   | PE        | U      | 10001  | 1025      |

## MonitorVirtualSummary

Collects global summary information on system utilization.

### Syntax

```
REPLACE FUNCTION SYSLIB.MonitorVirtualSummary (
) RETURNS TABLE (
  AMPAvgCPU FLOAT,
  AMPAvgDisk FLOAT,
  AMPAvgDiskIO FLOAT,
  HiCPUAMPUse FLOAT,
  HiDiskAMP FLOAT,
  HiDiskAMPIO FLOAT,
  HiCPUAMPNo SMALLINT,
  HiDiskAMPNo SMALLINT,
  HiDiskAMPIONo SMALLINT,
  HiCPUAMPProc INTEGER,
  HiDiskAMPProc INTEGER,
  HiDiskAMPIOProc INTEGER,
  LoCPUAMPUse FLOAT,
  LoDiskAMP FLOAT,
  LoDiskAMPIO FLOAT,
  LoCPUAMPNo SMALLINT,
  LoDiskAMPNo SMALLINT,
  LoDiskAMPIONo SMALLINT,
  LoCPUAMPProc INTEGER,
  LoDiskAMPProc INTEGER,
  LoDiskAMPIOProc INTEGER,
```

```

PEAvgCPU FLOAT,
HiCPUPEUse FLOAT,
LoCPUPEUse FLOAT,
HiCPUPENo SMALLINT,
LoCPUPENo SMALLINT,
HiCPUPEProc INTEGER,
LoCPUPEProc INTEGER,
SessionCnt FLOAT,
SesMonitorSys SMALLINT,
SesMonitorLoc SMALLINT,
ResLogging SMALLINT,
ResMonitor SMALLINT,
ReleaseNum CHAR(30) CHARACTER SET LATIN,
Version CHAR(32) CHARACTER SET LATIN
)
...
;

```

## Syntax Elements

### AMPAvgCPU

Average % CPU usage (CPUUse) of all online AMPs in the configuration.

Assuming  $n$  is the number of online AMPs in the configuration, AMPAvgCPU is computed from CPUUse data as:

$$(\text{CPUUse}_1 + \dots + \text{CPUUse}_n) / n$$

This value is NULL if certain conditions apply, see usage notes.

### AMPAvgDisk

Average physical disk usage (DiskUse) of all online AMPs in the configuration.

Assuming  $n$  is the number of online AMPs in the configuration, AMPAvgDisk is computed from DiskUse data as:

$$(\text{DiskUse}_1 + \dots + \text{DiskUse}_n) / n$$

This value is NULL if certain conditions apply, see usage notes.

### AMPAvgDiskIO

Average physical disk DiskReads and DiskWrites of all online AMPs in the configuration.

Assuming  $n$  is the number of online AMPs in the configuration, AMPAvgDiskIO is computed from DiskReads and DiskWrites data as:

$$(\text{DiskReads}_1 + \text{DiskWrites}_1 + \dots + \text{DiskReads}_n + \text{DiskWrites}_n) / n$$



This value is NULL if certain conditions apply, see usage notes.

#### **HiCPUAMPUse**

Highest CPUUse percentage currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **HiDiskAMP**

Highest DiskUse percentage currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **HiDiskAMPIO**

Highest DiskReads and DiskWrites value currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **HiCPUAMPNo**

Vproc number (VprocNo) of an AMP with CPUUse equal to the value reported as HiCPUAMPUse.

This value is NULL if certain conditions apply, see usage notes.

#### **HiDiskAMPNo**

Number of an AMP with DiskUse equal to the value reported as HiDiskAMP.

This value is NULL if certain conditions apply, see usage notes.

#### **HiDiskAMPIONo**

Number of an AMP with the highest DiskReads and DiskWrites equal to the value reported as HiDiskAMPIO.

This value is NULL if certain conditions apply, see usage notes.

#### **HiCPUAMPProc**

ID of the node currently responsible for managing the AMP reported as HiCPUAMPNo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

#### **HiDiskAMPProc**

ID of the node currently responsible for managing the AMP reported in HiDiskAMPNo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL when HiDiskAMPNo is NULL.

#### **HiDiskAMPIOProc**

ID of the node currently responsible for managing the AMP reported in HiDiskAMPIONo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL when HiDiskAMPIONo is NULL.

#### **LoCPUAMPUse**

Lowest CPUUse percentage currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **LoDiskAMP**

Lowest DiskUse percentage currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **LoDiskAMPIO**

Lowest DiskReads and DiskWrites number currently associated with any online AMP.

This value is NULL if certain conditions apply, see usage notes.

#### **LoCPUAMPNo**

Vproc number (VprocNo) of an AMP with CPUUse equal to the value reported as LoCPUAMPUse.

This value is NULL if certain conditions apply, see usage notes.

#### **LoDiskAMPNo**

Number of an AMP with DiskUse equal to the value reported as LoDiskAMP.

This value is NULL if certain conditions apply, see usage notes.

#### **LoDiskAMPIONo**

ID of an AMP with lowest DiskReads and DiskWrites equal to the value reported as LoDiskAMPIO.

This value is NULL if certain conditions apply, see usage notes.

**LoCPUAMPProc**

ID of the node currently responsible for managing the AMP reported as LoCPUAMPNo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**LoDiskAMPProc**

ID of the node currently responsible for managing the AMP reported as LoDiskAMPNo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL when LoDiskAMPNo is NULL.

**LoDiskAMPIOProc**

ID of the node currently responsible for managing the AMP reported as LoDiskAMPIONo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL when LoDiskAMPIONo is NULL.

**PEAvgCPU**

Average CPUUse for all online PEs in the configuration.

This value is NULL if certain conditions apply, see usage notes.

**HiCPUPEUse**

Highest CPUUse percentage currently associated with any online PE.

This value is NULL if certain conditions apply, see usage notes.

**LoCPUPEUse**

Lowest CPUUse percentage currently associated with any online PE.

This value is NULL when LoCPUPEUse is NULL.

**HiCPUPENo**

Vproc number (VProcNo) of a PE with CPUUse equal to the value reported as HiCPUPEUse.

This value is NULL if certain conditions apply, see usage notes.

**LoCPUPENo**

Vproc number (VProcNo) of a PE with CPUUse equal to the value reported as LoCPUPEUse.

This value is NULL if certain conditions apply, see usage notes.

**HiCPUPEProc**

ID of the node currently responsible for managing the PE reported in HiCPUPENo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL when HiCPUPENo is NULL.

**LoCPUPEProc**

ID of the node currently responsible for managing the PE reported as LoCPUPENo.

This value is computed as the module number within a cabinet plus the cabinet number times 10000. For example, a node #123 on cabinet #4 would return an INTEGER value of 40123.

This value is NULL if certain conditions apply, see usage notes.

**SessionCnt**

Total number of sessions currently logged onto the system. This value is usually equal to the sum of the SessLogCount values for all PEs.

This value is NULL if certain conditions apply, see usage notes.

**SesMonitorSys**

Sets the maximum acceptable age of collected session-level data in memory to the PM/API application or end user.

The global rate is the default collection rate for all MONITOR sessions. If the value is set to zero, the collection capability is disabled.

**SesMonitorLoc**

Sets the maximum acceptable age of collected session-level data in memory for an individual Monitor partition session that submits a MONITOR SESSION request.

This rate is initiated within a MONITOR session and may update session-level data within the system. If the value is zero, this allows SesMonitorSys to override the current local rate for that session.

**ResLogging**

Interval in seconds at which resource usage data is written to one or more active resource usage database tables.

**ResMonitor**

Interval in seconds at which all resource usage data is collected in memory for reporting via the PM/API.

**ReleaseNum**

Release number of the currently running database software (for example, 15.00.00.00).

This value is supplied by the database.

**Version**

Version number of the currently running database software (for example, 15.00.00.00).

This value is supplied by the database.

**Usage Notes**

The MonitorVirtualSummary function provides similar functionality to the PMPC MONITOR VIRTUAL SUMMARY request. For information about this interface, see [MONITOR VIRTUAL SUMMARY](#).

**Example: Using MonitorVirtualSummary**

```
select * from table (monitorvirtualsummary()) as t1;
```

\*\*\* Query completed. One row found. 35 columns returned.  
\*\*\* Total elapsed time was 2 seconds.

|                 |                       |
|-----------------|-----------------------|
| AMPAvgCPU       | 2.47375437427095E 001 |
| AMPAvgDisk      | 8.77770371604733E 000 |
| AMPAvgDiskIO    | 1.25500000000000E 003 |
| HiCPUAMPUse     | 2.48291951341443E 001 |
| HiDiskAMP       | 9.28178636893851E 000 |
| HiDiskAMPIO     | 1.27000000000000E 003 |
| HiCPUAMPNo      | 1                     |
| HiDiskAMPNo     | 1                     |
| HiDiskAMPIONo   | 1                     |
| HiCPUAMPProc    | 10001                 |
| HiDiskAMPProc   | 10001                 |
| HiDiskAMPIOProc | 10001                 |
| LoCPUAMPUse     | 2.46625562406266E 001 |
| LoDiskAMP       | 8.34860856523913E 000 |

```

LoDiskAMPIO      1.22300000000000E 003
LoCPUAMPNo       3
LoDiskAMPNo      3
LoDiskAMPIONo    0
LoCPUAMPProc     10001
LoDiskAMPProc    10001
LoDiskAMPIOProc  10001
PEAvgCPU         0.00000000000000E 000
HiCPUPEUse       0.00000000000000E 000
LoCPUPEUse       0.00000000000000E 000
HiCPUPENo       30719
LoCPUPENo       30719
HiCPUPEProc      10001
LoCPUPEProc      10001
SessionCnt       1.00000000000000E 000
SesMonitorSys    1
SesMonitorLoc    0
VprocLogging     60
VprocMonitor     60
ReleaseNum       16u.00.00.41
Version          16u.00.00.41_dr182707n

```

## MonitorWD

Returns the data in the second statement of the MONITOR WD response through an SQL interface consisting of UDFs.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorWD (
) RETURNS TABLE (
  WDIId INTEGER,
  PPIId SMALLINT,
  PGId SMALLINT,
  AGId SMALLINT,
  RelWgt SMALLINT,
  NumProcs INTEGER,
  VprType VARCHAR(4) CHARACTER SET LATIN,
  QWaitTime FLOAT,
  QWaitTimeMax FLOAT,
  CpuUserPct FLOAT,
  WorkMsgMaxDelay FLOAT,
  WorkTypeInuseMax INTEGER,
  WorkTimeInuseAvg FLOAT,

```

```

IODelay FLOAT,
IODelayTime FLOAT,
PhysicalRead FLOAT,
PhysicalReadKB FLOAT,
PhysicalWrite FLOAT,
PhysicalWriteKB FLOAT,
LogicalRead FLOAT,
LogicalReadKB FLOAT,
LogicalWrite FLOAT,
LogicalWriteKB FLOAT,
ExtraField1 FLOAT,
ExtraField2 FLOAT,
ExtraField3 FLOAT,
ExtraField4 FLOAT
VPId SMALLINT,
VPId SMALLINT,
WaitIO FLOAT,
WaitOther FLOAT,
CPURunDelay FLOAT,
IOCntSubmitted FLOAT,
IOKBSubmitted FLOAT,
IOCntCompleted FLOAT,
IOKBCompleted FLOAT,
IOCntCriticalSubmitted FLOAT,
IOKBCriticalSubmitted FLOAT,
DecayLevel1IO FLOAT,
DecayLevel2IO FLOAT,
DecayLevel1CPU FLOAT,
DecayLevel2CPU FLOAT,
TacticalExceptionIO FLOAT,
TacticalExceptionCPU FLOAT
)
...
;

```

## Syntax Elements

### PPId

This field is obsolete and returns a value of zero.

### PGId

This field returns the pWDId value.

**VprType**

Type of vproc:

- AMP
- PE
- MISC

**WDId**

WD ID. On SLES 11 or later systems, TASM Workloads rule is always enabled. On SUSE Linux Enterprise Server 11 or later systems, TASM Workloads rule is always enabled. For information on TASM rules, see *Teradata® Viewpoint User Guide*, B035-2206.

**AGId**

This field is obsolete and returns a value of zero.

**RelWgt**

This field is obsolete and returns a value of zero.

**NumProcs**

Average number of tasks of online nodes.

The field is the result of:

$$\text{NumProcs} = \text{SUM of } (\text{NumTasks-}i) / N$$

where:

- *NumTasks-i* is the number of tasks assigned to the WD at the end of the reporting period.
- *i* varies from 1 to *N*, where *N* is the number of online nodes.

The NumProcs field is the NumTasks field in the PM/API MONITOR WD request.

**QWaitTime**

Total wait time in milliseconds that work requests waited on an input queue before being serviced.

**QWaitTimeMax**

Maximum time in milliseconds that work requests waited on an input queue before being serviced.

The field is the result of:

$$\text{QWaitTimeMax} = \text{MAX } (\text{QWaitTimeMax-}i)$$

where:

- *QWaitTimeMax-i* is QWaitTimeMax in each online node.



- $i$  varies from 1 to  $N$ , where  $N$  is the number of online nodes.

### CPUUserPct

Weighted average of CpuUserPct of each node.

The field is the result of:

$$\text{CpuUserPct} = \text{Sum of } (\text{CpuUserPct-}i * \text{ScalingFactor-}i) / \text{Sum of } (\text{ScalingFactor-}i)$$

where:

- $\text{CpuUserPct-}i$  is calculated as:  

$$(\text{CPUUServAwt} + \text{CPUUServDisp} + \text{CPUUServMisc} + \text{CPUUExecAwt} + \text{CPUUExecDisp} + \text{CPUUExecMisc}) * 100 / (NCPUs * \text{Centisecs} * 10)$$

$NCPUs$  is the number of CPUs in the node.
- $i$  varies from 1 to  $N$ , where  $N$  is the number of online nodes.
- $\text{ScalingFactor-}i$  is the node CPU normalization factor in each node.

The CPU times are in milliseconds.

The Parser CPU times are included in the Dispatcher CPU times.

### WorkMsgMaxDelay

General indicator only. This field is result of the following calculation:

$$\text{WorkMsgMaxDelay} = \text{MAX } (\text{WorkMsgMaxDelay-}i)$$

where:

- $\text{WorkMsgMaxDelay-}i$  is calculated in each online node as:  

$$\text{WorkMsgsendDelayMax} + \text{WorkMsgReceiveDelayMax}$$
- $i$  varies from 1 to  $N$ , where  $N$  is the number of online nodes.

WorkMsgMaxDelay does not represent the subtotal of the same message on the send and receive side.

### WorkTypeInuseMax

Total of the AMP Worker Task (AWT) columns:

$$\text{WorkTypeInuseMax} = \text{MAX } (\text{WorkTypeInuseMax-}i)$$

where:

- $\text{WorkTypeInuseMax-}i$  is the sum of WorkTypeMax00 through WorkTypeMax15 in each node.
- $i$  varies from 1 to  $N$ , where  $N$  is the number of online nodes.

**WorkTimeInuseAvg**

Average number of AWTs used. This field is result of:

$\text{WorkTimeInuseAvg} = \text{SUM of } (\text{WorkTimeInuse-}i) / N$

where:

- *WorkTimeInuse- $i$*  is calculated in each online node as:  

$$\text{WorkTimeInuse} / (\text{Centisecs} * 10 * \text{NCPUs})$$
*NCPUs* is the number of CPUs in the node.
- $i$  varies from 1 to  $N$ , where  $N$  is the number of online nodes.

This value is available in the ResSpsView view as AwtUsedAvg.

**IODelay**

Number of I/Os that are delayed. This field is result of:

$\text{ProcBlksFsgRead} + \text{ProcBlksFsgWrite} + \text{ProcBlksFsgNIOs}$

**IODelayTime**

Total time the I/O is delayed for. This field is the result of:

$\text{ProcWaitFsgRead} + \text{ProcWaitFsgWrite} + \text{ProcWaitFsgNIOs}$

**PhysicalRead**

Number of physical reads performed for this period. This field is the result of:

$\text{FilePDdbAcqReads} + \text{FilePDdbPreReads} + \text{FilePCiAcqReads} + \text{FileSDbAcqReads} +$   
 $\text{FileSCiAcqReads} + \text{FileTJtAcqReads} + \text{FileAPtAcqReads} + \text{FilePCiPreReads} +$   
 $\text{FileSDbPreReads} + \text{FileSCiPreReads} + \text{FileTJtPreReads} + \text{FileAPtPreReads}$

**PhysicalReadKB**

Number of physical reads in KB performed for this period. This field is result of:

$\text{FilePDdbAcqReadKB} + \text{FilePDdbPreReadKB} + \text{FilePCiAcqReadKB} + \text{FileSDbAcqReadKB}$   
 $+ \text{FileSCiAcqReadKB} + \text{FileTJtAcqReadKB} + \text{FileAPtAcqReadKB} + \text{FilePCiPreReadKB} +$   
 $\text{FileSDbPreReadKB} + \text{FileSCiPreReadKB} + \text{FileTJtPreReadKB} + \text{FileAPtPreReadKB}$

**PhysicalWrite**

Number of physical writes performed for this period. This field is result of:

$\text{FilePDdbFWrites} + \text{FilePCiFWrites} + \text{FileSDbFWrites} + \text{FileSCiFWrites} + \text{FileTJtFWrites}$   
 $+ \text{FileAPtFWrites}$

**PhysicalWriteKB**

Number of physical writers in KB performed for this period. This field is result of:

FilePDbFWriteKB + FilePCiFWriteKB + FileSDbFWriteKB + FileSCiFWriteKB +  
FileTJtFWriteKB + FileAPtFWriteKB

**LogicalRead**

Number of logical reads performed for this period. This field is result of:

FilePDbAcqs + FilePDbPres + FilePCiAcqs + FileSDbAcqs + FileSCiAcqs + FileTJtAcqs +  
FileAPtAcqs + FilePCiPres + FileSDbPres + FileSCiPres + FileTJtPres + FileAPtPres

**LogicalReadKB**

Number of logical reads in KB performed for this period. This field is result of:

FilePDbAcqKB + FilePDbPresKB + FilePCiAcqKB + FileSDbAcqKB + FileSCiAcqKB +  
FileTJtAcqKB + FileAPtAcqKB + FilePCiPresKB + FileSDbPresKB + FileSCiPresKB +  
FileTJtPresKB + FileAPtPresKB

**LogicalWrite**

Number of logical writes performed for this period. This field is result of:

FilePDbDyRRels + FilePCiDyRRels + FileSDbDyRRels + FileSCiDyRRels +  
FileTJtDyRRels + FileAPtDyRRels

**LogicalWriteKB**

Number of logical writes in KB performed for this period. This field is result of:

FilePDbDyRRelKB + FilePCiDyRRelKB + FileSDbDyRRelKB + FileSCiDyRRelKB +  
FileTJtDyRRelKB + FileAPtDyRRelKB

**ExtraField1****ExtraField2****ExtraField3****ExtraField4**

This field is not currently used.

**VPIId**

Virtual partition ID.

**WaitIO**

Number of milliseconds tasks in WD waited for I/O over the reporting period.

WaitIO is updated when the wait for I/O is completed.

### **WaitOther**

Number of milliseconds tasks in WD waited for reasons other than I/O over the reporting period (for example, a task waiting for a message).

WaitOther is updated when wait is completed.

### **CPURunDelay**

Number of milliseconds tasks in the WD sat in the CPU runqueue waiting to run over the reporting period.

This data can be used in determining demand for the virtual partition and Workload Share Percent. The Workload Share Percent\* is a workload management method. If the CPU and I/O percentages for a virtual partition or WD are below their relative share values and the CPURunDelay values are low, there was insufficient demand to meet the share percentage. If the CPURunDelay values are high, higher tier SQL requests were allocated more resources so that there were insufficient resources remaining to allocate to SQL requests in this WD to meet its relative share.

A virtual partition divides a system so that a percentage of resources are allocated to a collection of workloads. A virtual partition can consist of WDs from all management methods.

### **IOSubmitted**

Number of I/Os submitted on behalf of this WD.

### **IOSubmittedKB**

KB of I/O submitted on behalf of this WD.

### **IOCompleted**

Number of AgeOut Now data blocks not to keep in memory (fsgcache) and to be written to disk.

### **IOCompletedKB**

KB of AgeOut Now data blocks not to keep in memory (fsgcache) and to be written to disk.

### **IOCriticalSubmitted**

Number of I/Os submitted with critical status. These I/Os execute at top priority instead of being based on the I/O priority of the SQL request.

**IOCriticalSubmittedKB**

KB of I/O submitted with critical status. These I/Os execute at top priority instead of being based on the I/O priority of the SQL request.

**DecayLevel1IO**

Number of times SQL requests in the WD hit decay level 1 due to I/O.

DecayLevel1IO is used for Timeshare WDs\*\* only.

**DecayLevel2IO**

Number of times SQL requests in the WD hit decay level 2 due to I/O.

DecayLevel1IO is used for Timeshare WDs\*\* only.

**DecayLevel1CPU**

Number of times SQL requests in the WD hit decay level 1 due to CPU.

DecayLevel1IO is used for Timeshare WDs\*\* only.

**DecayLevel2CPU**

Number of times SQL requests in the WD hit decay level 2 due to CPU.

DecayLevel1IO is used for Timeshare WDs\*\* only.

**TacticalExceptionIO**

Number of times SQL requests in the WD hit a tactical per-node exception due to I/O.

An exception, used only for Tactical WDs, is created for each Tactical WD\*\*\*.

**TacticalExceptionCPU**

Number of times SQL requests in the WD hit a tactical per-node exception due to CPU.

TacticalExceptionCPU is used for Tactical WDs\*\*\* only.

\* The Workload Share Percent Management Method workload is assigned a proportion of the resources that are available after allocations have been made for tactical workloads. The percentage of resources is divided equally between all requests running in the WD. For example, if the Workload Share Percent is 5% and there are five SQL requests, each SQL request will get 1% of the share resources. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

\*\* The Timeshare Workload Management Method workload can be assigned to one of four stepped access levels, Top, High, Medium, or Low. The higher access levels are given larger access rates than the lower levels. For example, an SQL request assigned to a Timeshare WD with a Top access level, which has an

access rate of 8, would receive eight times the amount of resources than an SQL request assigned to a Low access level.

Timeshare workloads are assigned resources remaining after all allocations have been made for tactical and Workload Share Percent workloads. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

\*\*\* The Tactical Workload Management Method workload yields the fastest available response time and executes at the highest tier, preempting all resource needs of other tiers. This method is well suited for critical, short-running queries that require fast response times. For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

## Usage Notes

The MonitorWD function provides similar functionality to the PMPC MONITOR WD request. For information about this interface, see [MONITOR WD](#).

For information on the resource usage tables and columns described in the field calculations below, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

### Example: Using MonitorWD

```
SELECT * FROM TABLE (MonitorWd()) AS t2 where vpid=102 and vprtype='amp';
```

```
*** Query completed. One row found. 43 columns returned.
```

```
*** Total elapsed time was 1 second.
```

|                  |                       |
|------------------|-----------------------|
| WDId             | 0                     |
| PPIId            | 0                     |
| PGId             | 254                   |
| AGId             | 0                     |
| RelWgt           | 0                     |
| NumProcs         | 454                   |
| VprType          | AMP                   |
| QWaitTime        | 0.00000000000000E 000 |
| QWaitTimeMax     | 0.00000000000000E 000 |
| CpuUserPct       | 0.00000000000000E 000 |
| WorkMsgMaxDelay  | 0.00000000000000E 000 |
| WorkTypeInUseMax | 0                     |
| WorkTimeInUseAvg | 0.00000000000000E 000 |
| IODelay          | 1.60000000000000E 001 |
| IODelayTime      | 1.19968000000000E 005 |
| PhysicalRead     | 0.00000000000000E 000 |
| PhysicalReadKB   | 0.00000000000000E 000 |
| PhysicalWrite    | 1.60000000000000E 001 |
| PhysicalWriteKB  | 6.56000000000000E 002 |
| LogicalRead      | 8.00000000000000E 000 |
| LogicalReadKB    | 2.56000000000000E 002 |

|                       |                   |     |
|-----------------------|-------------------|-----|
| LogicalWrite          | 8.00000000000000E | 000 |
| LogicalWriteKB        | 2.56000000000000E | 002 |
| ExtraField1           | 0.00000000000000E | 000 |
| ExtraField2           | 0.00000000000000E | 000 |
| ExtraField3           | 0.00000000000000E | 000 |
| ExtraField4           | 0.00000000000000E | 000 |
| VPIId                 | 102               |     |
| WaitIO                | 5.80000000000000E | 001 |
| WaitOther             | 7.78500000000000E | 005 |
| CPURunDelay           | 2.98000000000000E | 002 |
| IOSubmitted           | 1.60000000000000E | 001 |
| IOSubmittedKB         | 1.48000000000000E | 002 |
| IOCompleted           | 1.60000000000000E | 001 |
| IOCompletedKB         | 1.48000000000000E | 002 |
| IOCriticalSubmitted   | 0.00000000000000E | 000 |
| IOCriticalSubmittedKB | 0.00000000000000E | 000 |
| DecayLevel1IO         | 0.00000000000000E | 000 |
| DecayLevel2IO         | 0.00000000000000E | 000 |
| DecayLevel1CPU        | 0.00000000000000E | 000 |
| DecayLevel2CPU        | 0.00000000000000E | 000 |
| TacticalExceptionIO   | 0.00000000000000E | 000 |
| TacticalExceptionCPU  | 0.00000000000000E | 000 |

## MonitorWDRate

Returns the collection rate, number of nodes with at least one online AMP, and number of nodes with at least one online PE.

### Syntax

```

REPLACE FUNCTION SYSLIB.MonitorWDRate (
) RETURNS TABLE (
    SampleRate SMALLINT,
    AMPNodes SMALLINT,
    PENodes SMALLINT
)
...
;

```

### Syntax Elements

#### SampleRate

Number of seconds of the collection period.

**AMPNodes**

Number of nodes with at least one online AMP.

**PENodes**

Number of nodes with at least one online PE.

**Example: Using MonitorWDRate**

```
SELECT * FROM TABLE (MonitorWDRate()) AS t2;
*** Query completed. One row found. 3 columns returned.
*** Total elapsed time was 1 second.
SampleRate      600  AMPNodes      1  PENodes      1
```

**SetResourceRate**

Sets either the ResMonitor or ResLogging rate.

This function returns the previous collection rate value in seconds.

**Syntax**

```
REPLACE FUNCTION SYSLIB.SetResourceRate (
  SampleRate INTEGER,
  LogChange VARCHAR(1),
  VprocChange VARCHAR(1)
) RETURNS INTEGER
...
;
```

**Syntax Elements*****SampleRate***

Value of the collection rate. This field is used either to collect resource data or to log resource data to the resource usage tables.

You can specify one of the following:

- The ResMonitor rate value if you want to change the resource monitoring rate.
- The ResLogging rate value if you want to change the resource logging rate.

The value should be an integral divisor of 3600.

- Zero to turn off the resource collection or logging.



**LogChange**

Indicator of whether this rate applies to the ResLogging or ResMonitor rate:

- Y or y = ResLogging rate
- N, n, NULL, or blank = ResMonitor rate

**VprocChange**

This field is obsolete.

**Usage Notes**

The SetResourceRate function provides similar functionality to the PMPC SET RESOURCE RATE request. For information about this interface and the ResMonitor and ResLogging rates, see [SET RESOURCE RATE](#).

**Example: Using SetResourceRate**

This example uses SetResourceRate to set the ResMonitor rate.

```
SELECT SetResourceRate(60, 'N', 'N');
*** Query completed. One row found. One column returned.
*** Total elapsed time was 5 seconds.
SetResourceRate(60,'N','N')
-----
                        600
```

**SetSessionAccount**

Changes the account string for the session or for the request.

This function returns the old account string.

**Syntax**

```
REPLACE FUNCTION SetSessionAccount (
  HostIdIn SMALLINT,
  SessionNoIn INTEGER,
  NewAcctString TD_ANYTYPE,
  EntireSession VARCHAR(1) CHARACTER SET LATIN
) RETURNS VARCHAR(128) CHARACTER SET UNICODE
  ...
;
```

## Syntax Elements

### *HostIdIn*

Logical ID of a host (or client) with sessions logged on.

### *SessionNoIn*

Number of the specified session.

### *NewAcctString*

Account string for the session or request.

### *EntireSession*

Indicator of how the new account or priority affects requests for a specified session.

If you specify Y or y, the change applies to all current and future requests for a specified session. If no requests or steps are executing, the new account or priority takes effect at the next request, and the DBC.SessionTbl table reflects the new account or priority for the current session.

If you specify NULL, blank, N, or n, the change applies only to the current request for the specified session. If no request is executing, the next request for the specified session has the old account/priority.

## Usage Notes

The SetSessionAccount function provides similar functionality to the PMPC SET SESSION ACCOUNT request. For information about this interface, see [SET SESSION ACCOUNT](#).

### Example: Using SetSessionAccount with MonitorSession

```
BTEQ -- Enter your DBC/SQL request or BTEQ command:
SELECT SetSessionAccount(Hostid, sessionno, 'Accountx', 'Y')
FROM TABLE (MonitorSession(1, 'twmuser3', 0)) AS t1;
SELECT SetSessionAccount(Hostid, sessionno, 'Accountx', 'Y')
FROM TABLE (MonitorSession(1, 'twmuser3', 0)) AS t1;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
SetSessionAccount(HostId, SessionNo, 'Accountx', 'Y')
-----
ACCOUNT3
```

## Example: Using SetSessionAccount

```
BTEQ -- Enter your DBC/SQL request or BTEQ command:
SELECT SetSessionAccount(1, 4461, 'Account3','y');
SELECT SetSessionAccount(1, 4461, 'Account3','y');
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
SetSessionAccount(1,4461,'Account3','y')
-----
ACCOUNTX
```

## SetSessionRate

Sets the global rates for updating session-level statistics in memory.

This function returns the previous collection rate value in seconds.

### Syntax

```
REPLACE FUNCTION SetSessionRate (
    SampleRate INTEGER
) RETURNS INTEGER
...
;
```

### Syntax Elements

#### *SampleRate*

Sets the global rate. The value ranges from 1 to 3600 seconds. If the rate is set to zero an error is reported.

The collection rate applies to global queries.

### Usage Notes

There is no option equivalent to the PMPC SET SESSION RATE request *Local\_Change* = Y option because when using the SetSessionRate function, there is no local MONITOR PARTITION in which to apply the rate to.

You can set the SetSessionRate function to a global (SesMonitorSys) rate. That is, a collection rate at which session-level statistics are collected in memory. This rate is returned in the SesMonitorSys data value by the MonitorVirtualSummary function.

The SetSessionRate function provides similar functionality to the PMPC SET SESSION RATE request. For information about this interface, see [SET SESSION RATE](#).

**Example: Using SetSessionRate**

```
SELECT SetSessionRate(600);  
*** Query completed. One row found. One column returned.  
*** Total elapsed time was 1 second.  
SetSessionRate(600)  
-----  
60
```

# Teradata Dynamic Workload Management APIs: PM/APIs

The following discusses how to manage TASM operations through two types of interfaces:

- [PM/APIs](#)
- [Open APIs \(SQL Interfaces\)](#)

Before using the Teradata Dynamic Workload Management APIs, you may want to familiarize yourself with the following topics:

- [Teradata Dynamic Workload Management API Features](#)
- [Examples Using Open APIs](#)

## PM/APIs

This section describes the Teradata Dynamic Workload Management interfaces that use CLIV2 or the Teradata JDBC Driver. These interfaces are referred to as PM/API requests.

For details on using the Teradata JDBC Driver requests, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

Before using the Teradata Dynamic Workload Management PM/APIs, read the following topic on the impact of object name length on PM/APIs requests.

### Impact of Object Name Length on PM/API Requests

Object names contain 128 Unicode characters; however, to use object names of 128 characters, applications must use monitor software version 10 or later.

| If your custom application uses monitor software version ... | The object field in the input area can be ...                                                                                                                                                                                           |
|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8 or earlier                                                 | CHAR(30). A fixed parcel field size of 30 bytes. The names must be padded with spaces.<br>For more information on monitor software version 8 or earlier, see previous releases of <i>Workload Management API: PM/API and Open API</i> . |
| 9                                                            | VARCHAR(120). A parcel field size of up to 120 bytes in variable length.<br>For more information on monitor software version 9, see <i>Application Programming Reference: Workload Management</i> for Teradata Database 14.0.           |
| 10 or later                                                  | VARCHAR(512). A parcel field size of up to 512 bytes in variable length in host character set format.                                                                                                                                   |

Depending on the version of your custom application, some of the PM/API request output fields for object names may be truncated. For example, if the output field is an object name of 100 bytes in length when

converted to host character set format, and you are using a PM/API request with monitor software version 8 or earlier, the database name will be truncated to fit the fixed parcel field size of 30 bytes.

## EVENT STATUS

Returns information about the current status of all event-related constructs, such as events through states, and information related to the current state.

### Input Data

| Element             | Data Type             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte             | BYTE                  | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode. |
| <i>mon_ver_id</i>   | SMALLINT,<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |
| <i>Request Type</i> | SMALLINT,<br>NOT NULL | 1 = ALL. This option returns the status of all events, expressions, SysCons, and OpEnvs defined in the Teradata Viewpoint Workload Designer portlet.<br>2 = CURRENT. This option includes the status of those events and expressions that are related to the currently enforced health condition (SysCon) and planned environment (OpEnv).                                                                                                                    |
| <i>ID Mapping</i>   | SMALLINT              | Mapping information returned in the output, such as the names and IDs defined within the current rule set.<br>If a value of 1 is returned, mapping record types are included. Otherwise, no mapping information is returned.                                                                                                                                                                                                                                  |

### Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## CLlv2 Response Parcels

The following table lists information about the parcels.

| Parcel Sequence | Flavor | Field Length                                                                                                    | Comments and Key Parcel Body Fields                                                                                                                                                                                                                            |
|-----------------|--------|-----------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8      | 18 to 273                                                                                                       | StatementNo = 1<br>ActivityCount = Not applicable<br>ActivityType = 170 (PCLEVENTSTATUSSTMT)                                                                                                                                                                   |
| DataInfo        | 71     | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                                                                                                                              |
| Record          | 10     | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This is the only record returned.<br>This record contains an indicator for each category indicating its status after processing this request (0=active, 1=inactive) |
| EndStatement    | 11     | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                                                                                                   |
| EndRequest      | 12     | 4                                                                                                               | None                                                                                                                                                                                                                                                           |

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Response

### Note:

Each of the statement types described below correspond to a `ResultSet` returned by the Teradata JDBC Driver, and each statement type field corresponds to a `ResultSet` column. For more information on `ResultSets`, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The EVENT STATUS request returns the status of all items that make up either the CURRENT or ALL states defined in the current rule set.

The output is divided into three statements:

- The first statement provides configuration overview information.
- The second statement provides a series of records that represent the status of events, expressions and states in the system. Only events that are referenced in Expressions that have an action are included in this statement. To optimize the interface, this information is provided using the internal ID of the various events, expressions and states.
- The third statement, which is optional, provides the mapping of internal IDs to names.

## Statement 1

The first statement returns information about the current configuration.

| Field/Column Name | Data Type           | Description                                       |
|-------------------|---------------------|---------------------------------------------------|
| Config ID         | INTEGER<br>NOT NULL | Configuration ID of the current rule set.         |
| Config Name       | VARCHAR (30)        | Name of the current rule set, blank filled.       |
| Date              | DATE                | Date when this configuration was first activated. |
| Time              | FLOAT               | Time when this configuration was first activated. |

Use this first statement to determine if a new configuration has been placed into effect. When a new configuration is placed into effect, the mapping record type should be requested to provide the mapping of object IDs to object names.

## Statement 2

The second statement returns a series of records that provide the status of events, expressions, SysCons and OpEnvs in the system. The information returned shows the relationship between an Event, the Expression that references that Event, and the SysCon or OpEnv that is affected by that Expression.

There is one record returned for every Event that is referenced in the associated Expression. When CURRENT is requested only a single Expression for the CURRENT SysCon and OpEnv that caused that SysCon or OpEnv to be defined as enforced is returned. When ALL is requested, ALL expressions for each SysCon or OpEnv are returned. In addition, the ALL interface returns those expressions which do not have an action related to a SysCon or OpEnv. These are referred to as Notify Only Expressions.

The expression and owning SysCon or OpEnv ID fields are repeated until all events are included. If events are repeated within multiple expressions, those events are reported within each of those expressions. For the case of the ALWAYS OpEnv and the NORMAL SysCon, there is no expression associated. Statements for these two items have a value of zero for the Event and Expression ID fields.

| Field/<br>Column Name | Data Type           | Description                                                                                                                                                                                                                                                                                                      |
|-----------------------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EventId               | INTEGER<br>NOT NULL | Internal ID of the Event. A value of zero indicates there is no event for this record entry. There are no expressions for the Normal SysCon, Always OpEnv, and a State.<br>Events are reported in the order they appear within the associated expression. Only events in the associated expression are included. |
| EventStatus           | SMALLINT            | Status of the event: <ul style="list-style-type: none"> <li>• 0 = Inactive</li> <li>• 1 = Active</li> </ul>                                                                                                                                                                                                      |
| EventActiveDate       | DATE                | Date when EventStatus was last changed. This is the date when EventStatus was set to Inactive or Active.                                                                                                                                                                                                         |



| Field/<br>Column Name  | Data Type           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Event Active Time      | FLOAT               | Time when EventStatus was last changed. This is the time when EventStatus was set to Inactive or Active.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ExpressionId           | INTEGER<br>NOT NULL | Internal ID of the Expression.<br>A value of zero indicates there is no expression for this record entry. There are no expressions for the Normal SysCon, Always OpEnv, and a State.<br>For the CURRENT state request, the first TRUE expression for the associated State is included. When there are multiple true expressions for a state only the first one processed is reported. The order of processing is consistent within a rule set and, therefore, the same True expression is reported on every request.                                                                                                                                                                                                                                                                                                                       |
| Expression Status      | SMALLINT            | Current status of the Expression: <ul style="list-style-type: none"> <li>• 0 = Inactive</li> <li>• 1 = Active</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Expression Active Date | DATE                | Date when Expression Status was last changed. This is the date when Expression Status was set Active or Inactive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Expression Active Time | FLOAT               | Time Expression Status was last changed. This is the time when Expression Status was set Active or Inactive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Record Type            | SMALLINT            | Type of information contained in this record: <ul style="list-style-type: none"> <li>• 0 = SysCon</li> <li>• 1 = OpEnv</li> <li>• 2 = Notify Only Expression. This option indicates the remaining fields are not applicable as only the Event or Expression relationship is reported in this statement.</li> <li>• 3 = State. This option indicates the current active State. This is the result of the current SysCon and OpEnv. For this Record Type, the Event and Expression fields above are not valid.</li> </ul> For both the CURRENT and ALL requests, the order in which State information is returned is as follows: <ul style="list-style-type: none"> <li>• SysCon</li> <li>• OpEnv</li> <li>• State (Returns only the Enforced State)</li> </ul> <b>Note:</b><br>Notify Only Expression is only returned for the ALL request. |
| Id                     | INTEGER<br>NOT NULL | Internal ID of either a SysCon or an OpEnv depending on the Record Type specified.<br>The value is zero for Notify Only Expression record types. Notify Only Expression records represent Expressions that do not have an action to activate a Syscon or OpEnv. These Expressions are not associated with a State.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Status                 | SMALLINT            | Current status of the Syscon or OpEnv: <ul style="list-style-type: none"> <li>• 0 = Inactive</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| Field/<br>Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |           | <ul style="list-style-type: none"> <li>• 1 = Active</li> <li>• 2 = Active and Enforced</li> </ul>                                                                                                                                                                                                                                                                                                                                                                         |
| Due To Duration       | INTEGER   | <p>Indicator that the corresponding SysCon is being considered as TRUE or is TRUE:</p> <ul style="list-style-type: none"> <li>• 1 = SysCon is Active if the corresponding SysCon is being considered as TRUE due to minimum duration (MinDuration) value being in affect.</li> <li>• 0 = SysCon is Inactive if the Syscon is TRUE based on the defined expressions.</li> </ul> <p><b>Note:</b><br/>This field is valid only when SysCon is the Record Type specified.</p> |
| Active Date           | DATE      | Date when the status of the Record Type was last changed. This is the date when the SysCon or OpEnv Status was set Active or Inactive.                                                                                                                                                                                                                                                                                                                                    |
| Active Time           | FLOAT     | Time representing when the status of the Record Type was last changed. This is the time when the SysCon or OpEnv Status was set Active or Inactive.                                                                                                                                                                                                                                                                                                                       |

The date or time for each item (for example, Event, Expression, or State) shows when the status that is represented occurred.

**Note:**

The “State” column in the examples below refers to the “Expression Status” field.

| EvtID | Status | Date/Time     | ExpID | State | Date/Time     | RecType |
|-------|--------|---------------|-------|-------|---------------|---------|
| Id    | Status |               |       |       |               |         |
| 110   | 1      | 8:00AM 1/2/06 | 220   | 1     | 8:00AM 1/2/06 | 0 300 1 |
| 120   | 0      | 8:15AM 1/2/06 | 220   | 1     | 8:15AM 1/2/06 | 0 300 1 |
| 130   | 1      | 8:00AM 1/2/06 | 320   | 1     | 8:00AM 1/2/06 | 1 400 1 |
| 140   | 1      | 8:15AM 1/2/06 | 320   | 1     | 8:15AM 1/2/06 | 1 400 1 |

This statement shows two records representing a Single expression (220) that is defined with two Events (110 and 120). In this example, 110 is true and 120 is false. The original expression must be referenced by the user to determine why these event values resulted in the Expression being true. That is, these events are part of a logical expression whose structure is not being reported here. Another expression (320) is shown that is made up of two events (130 and 140). This expression is responsible for the OpEnv of 400 being in enforced.

| EvtID | Status | Date/Time     | ExpID | State | Date/Time     | RecType | Id  | Status |
|-------|--------|---------------|-------|-------|---------------|---------|-----|--------|
| 0     | 0      | 8:00AM 1/2/06 | 0     | 0     | 8:00AM 1/2/06 | 0       | 100 | 1      |
| 0     | 0      | 8:15AM 1/2/06 | 0     | 0     | 8:15AM 1/2/06 | 1       | 200 | 1      |

This record has both event and expression IDs of zero. This is the case for both the NORMAL SysCon and the ALWAYS OpEnv which have no expressions associated with them. The Date/Time fields are not valid since they are not associated with an Event or Expression but are only shown as placeholders.

For the CURRENT request, the first Active expression for a SysCon/OpEnv is returned. Within this Active (true) expression, all events, both Active and Inactive, are reported. The interface reports the events in a consistent order for a given expression.

For the ALL request, the order of reporting is that all SysCons are returned, followed by all OpEnvs, then all Expressions not associated with a SysCon or OpEnv State. These are referred to as Notify Only Expressions. Within a SysCon or OpEnv, all expressions for that State are reported before reporting the next SysCon or OpEnv. There is no order defined for the expressions reported under a SysCon/OpEnv but the interface ensures that expressions are returned in a consistent order.

The following example shows the reporting of three SysCons (300, 310 and 400), three OpEnvs (500, 530, and 540), and one Notify Only Expression (329) made up of one event (199). This Notify Only Expression does not have an ID for the Record Type (2). Although not shown in the example, only one SysCon and one OpEnv reported have the enforced value returned (a Status of 2). You must independently use the TDWM database State table to make the association to a State using the returned SysCon and OpEnv.

---

**Note:**

The “State” column in the example below refers to the “Expression Status” field.

| EvtID   | Status | Date/Time     | ExpID | State | Date/Time     |   |     |   |  |
|---------|--------|---------------|-------|-------|---------------|---|-----|---|--|
| RecType | Id     | Status        |       |       |               |   |     |   |  |
| 110     | 1      | 8:00AM 1/2/06 | 220   | 0     | 8:00AM 1/2/06 | 0 | 300 | 0 |  |
| 120     | 0      | 8:15AM 1/2/06 | 220   | 1     | 8:15AM 1/2/06 | 0 | 310 | 1 |  |
| 130     | 1      | 8:00AM 1/2/06 | 223   | 1     | 8:00AM 1/2/06 | 0 | 400 | 1 |  |
| 140     | 1      | 8:15AM 1/2/06 | 223   | 1     | 8:15AM 1/2/06 | 0 | 400 | 1 |  |
| 150     | 1      | 8:00AM 1/2/06 | 244   | 0     | 8:00AM 1/2/06 | 1 | 500 | 1 |  |
| 160     | 0      | 8:15AM 1/2/06 | 220   | 1     | 8:15AM 1/2/06 | 1 | 500 | 1 |  |
| 170     | 1      | 8:00AM 1/2/06 | 320   | 1     | 8:00AM 1/2/06 | 1 | 540 | 1 |  |
| 180     | 1      | 8:15AM 1/2/06 | 320   | 1     | 8:15AM 1/2/06 | 1 | 530 | 1 |  |
| 199     | 1      | 8:15AM 1/2/06 | 329   | 1     | 8:15AM 1/2/06 | 2 | 0   | 0 |  |

---

### Statement 3

The third statement, if requested, provides a fully enumerated list of all objects that make up the system state. Some of these objects may not be represented in any expressions in the state definition. There are no duplications of objects in this statement and there is no correlation made between objects.

Since this information does not change frequently this mapping information only needs to be requested when a new configuration is made active by the administrator.

| Field/<br>Column<br>Name | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|--------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Object Type              | SMALLINT             | Type of object: <ul style="list-style-type: none"> <li>• 0 = Event</li> <li>• 1 = Expression</li> <li>• 2 = SysCon</li> <li>• 3 = OpEnv</li> <li>• 4 = State</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Object Id                | INTEGER,<br>NOT NULL | Internal object ID for the object type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Object Name              | VARCHAR<br>(30)      | Name of the object.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Object Status            | SMALLINT             | Current status of the object: <ul style="list-style-type: none"> <li>• 0 = Inactive</li> <li>• 1 = Active</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Precedence/<br>Severity  | INTEGER              | Indicator of the Precedence or Severity of the SysCon or OpEnv object types only: <ul style="list-style-type: none"> <li>• SysCon = Indicates the Severity of the SysCon.</li> <li>• OpEnv = Indicates the Precedence of the OpEnv.</li> </ul> <p><b>Note:</b><br/>This field does not apply if this is an Event.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Date                     | DATE                 | Date of the current object status (see Object Status).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Time                     | FLOAT                | Time of the current object status (see Object Status).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| EventKind                | INTEGER              | Indicator of the specific event: <ul style="list-style-type: none"> <li>• 1 = User Defined SysCon. The user-defined system condition.</li> <li>• 2 = User Defined OpEnv. The user-defined operating environment.</li> <li>• 3 = Time Period. The time range.</li> <li>• 4= Fatal AMPs. The number of AMPs with the status of FATAL.</li> <li>• 5 = Fatal PEs. The number of PEs with the status of FATAL.</li> <li>• 6 = Fatal GTWs. The number of Gateways with the status of FATAL.</li> <li>• 7 = Nodes Down. The % of nodes down within a clique.</li> <li>• 8 = Minimum available AWTs. The number of in-use AMP AWTs.</li> <li>• 9 = AMPs In Flow Control. The number of AMPs in flow control.</li> <li>• 10 = Average System CPU. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables and compared to the threshold defined by the user in this event. See <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099 for information.</li> <li>• 11 = System CPU Skew. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables. For more</li> </ul> |

| Field/<br>Column<br>Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |           | <p>information, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</p> <ul style="list-style-type: none"> <li>• 12 = WD CPU %. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables. For more information, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</li> <li>• 13 = WD SLG Response Time. The % of request in this WD that have met the defined Response Time SLG.</li> <li>• 14 = WD SLG Throughput. The % of request in this WD that have met the defined Throughput SLG.</li> <li>• 15 = WD Arrivals. The number of new requests for this WD.</li> <li>• 16 = WD Active Request. The number of currently active requests in this WD.</li> <li>• 17 = WD Delay Queue Depth. The number of requests on the Delay Queue for this WD.</li> <li>• 18 = WD Delay Query Time. The max time a request has been delayed in this WD.</li> <li>• 19 = WD AWT Wait Time. The max time a request from this WD was on an AMP mailbox waiting for an AWT.</li> <li>• 20 = TwmFlexAvailableAWTsEvent. The number of AWTs that are available for work.</li> <li>• 21 = TwmFlexAvgCpuEvent. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables and compared to the threshold defined by the user in this event. For more information, see <i>Teradata Vantage™ - Resource Usage Macros and Tables</i>, B035-1099.</li> </ul> |
| EventClass               | INTEGER   | <p>This field is only valid for Object Type of Event and Expression. The values are:</p> <ul style="list-style-type: none"> <li>• 1 = OpEnv</li> <li>• 2 = SysCon</li> <li>• 3 = FlexThrtl</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

The following example shows the relationship between the Event Id (110) in statement 2 and the Object Name (AMPDownEvent) in statement 3.

The data returned looks similar to this:

```
Object type = 0
Object Id    = 110
Object name  = AMPDownEvent
Object status = 0
Precedence/Severity=
Date = 04/04/07
```

```
Time = 02:30:06
EventKind = 9
```

The following is the order of objects returned for statement 3:

1. All the rows for the Events
2. A row for a SysCon and all Expressions that are part of that Syscon
3. A row for the next SysCon and its subordinate Expressions until all SysCons are reported
4. A row for a OpEnv and all Expressions that are part of that OpEnv
5. A row for the next OpEnv and its subordinate Expressions until all OpEnvs are reported
6. All the rows for the States
7. All the rows for the Notify Only Expressions

## PERFGROUPS

Returns information about the Resource Partitions (RPs) and Performance Groups (PGs) active in the system GDO file.

### Note:

On database systems running SLES 11 or later systems, this function is obsolete and returns an error.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte           | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode. |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |

### Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

A Resource Partition is a subset of database request-processing resources. The default Resource Partition represents 100% of the system resources. Each Resource Partition contains a number of Performance Groups that have access to the resources available to the Resource Partition. You can define additional Resource Partitions and populate them with Performance Groups.

A Performance Group is a collection users that have access to the resources of the Resource Partition to which they belong. You can assign users to Performance Groups using a MODIFY USER statement. When users log on they can enter the Performance Group as part of the account string. If a user is not assigned to a Performance Group or does not enter one as part of the account string at logon, the system uses the default Performance Group for the session.

There are two types of Resource Partitions and Performance Groups:

- Those that are set up using the Priority Scheduler function in the schmon utility

---

### Note:

On SLES 11 systems, Priority Scheduler is managed by TASM, and is configured using the Teradata Viewpoint workload management portlets. For more information on those portlets, see *Teradata® Viewpoint User Guide*, B035-2206.

---

- Those that are part of a Teradata dynamic workload management software WD

Only one of these types can be active at any time.

To learn more about Resource Partitions and Performance Groups, see *Teradata Vantage™ - Database Utilities*, B035-1102 and *Teradata Vantage™ - Database Administration*, B035-1093.

The PERFGROUPS request returns the set of Resource Partitions and Performance Groups that are currently active.

- When TASM Workloads are not enabled, the returned set of Resource Partitions or Performance Groups are those defined by the schmon utility.
- When TASM Workloads are enabled, the returned set of Resource Partitions or Performance Groups are those defined by the Teradata Viewpoint Workload Designer portlet.

## CLIV2 Response Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields      |
|-----------------|---------------|----------------|--------------------------------------|
| Success         | 8             | 18 to 273      | StatementNo = 1<br>ActivityCount = 5 |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                      | Comments/Key Parcel Body Fields                                                                                                                                                                                                                                                       |
|-----------------|---------------|---------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |               |                                                                                                                     | ActivityType = 133 (PCLTWMPERFGROUPSSTMT)                                                                                                                                                                                                                                             |
| DataInfo        | 71            | 6 to 64100                                                                                                          | Optional; this parcel is present if request was IndicData parcel.                                                                                                                                                                                                                     |
| Record          | 10            | <ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is returned in record or indicator mode. One record is returned for each RP slot in the system GDO. The unused slots are padded with blanks. The format of this Resource Partition Record parcel is described below.               |
| EndStatement    | 11            | 6                                                                                                                   | StatementNo = 1                                                                                                                                                                                                                                                                       |
| Success         | 8             | 18 to 273                                                                                                           | StatementNo = 2<br>ActivityCount = 40<br>ActivityType = 133 (PCLTWMPERFGROUPSSTMT)                                                                                                                                                                                                    |
| DataInfo        | 71            | 6 to 64100                                                                                                          | Optional; this parcel is present if request was IndicData parcel.                                                                                                                                                                                                                     |
| Record          | 10            | <ul style="list-style-type: none"> <li>• 5 to 64100 (record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is returned in record or indicator mode. One record is returned for each Performance Group slot in the system GDO. The unused slots are padded with blanks. The format of this Performance Group Record parcel is described below. |
| EndStatement    | 11            | 6                                                                                                                   | StatementNo = 2                                                                                                                                                                                                                                                                       |
| EndRequest      | 12            | 4                                                                                                                   | None                                                                                                                                                                                                                                                                                  |

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver.

For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The Resource Partition Record parcel contains the *Resource Partition Name* field. This field returns the name of the Resource Partition. The names are padded with blanks to fill the entire field.

*Resource Partition Name* is a VARCHAR data type with a maximum variable length of 16 characters.

The following table describes the format of the Performance Group Record parcel.



| Field/Column Name        | Data Type    | Description                                                                                                                                                                      |
|--------------------------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Performance Group Name   | VARCHAR (16) | Name of the Performance Groups. The names are padded with blanks to fill the entire field.<br><br><b>Note:</b><br>The unused Performance Groups have names which are all blanks. |
| Resource Partition Index | SMALLINT     | Index that associates this Performance Group back to one of the Resource Partitions named in a previous Resource Partition record.                                               |

### Sample Input - CLIV2 Request

The following example illustrates how the parcels for a PERFGROUPS request, built by CLIV2, look when sent to the database server when the *mon\_ver\_id* value is not checked. In this example, the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body       |            |
|--------|------|--------|------------|------------|
| Num    | Name | Bytes  | Field      | Value      |
| 0001   | Req  | 17     | Request    | PERFGROUPS |
| 0004   | Resp | 6      | BufferSize | 64000      |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

The following examples show typical data sets returned in character text format for the PERFGROUPS request.

The PERFGROUPS request you submit may return information in a different format.

#### Note:

The original system Performance Groups are saved while the Teradata Viewpoint Workload Designer portlet settings are used and are reactivated if TASM Workloads are disabled.

### Sample Output - Using PERFGROUPS to Show User-defined Priority Scheduler Settings

This example shows a representative set of user-defined (using *schmon*) Priority Scheduler settings that are used by the system when TASM Workloads are disabled.

```
Submitting request TDWM PERFGROUPS;
Resource Partition items: 5
    Resource Partition Name : Default
    Resource Partition Name : RP1
    Resource Partition Name : standard
    Resource Partition Name :
    Resource Partition Name :
Performance Group items: 40
    RP#: 0, PG-Name: L
    RP#: 0, PG-Name: M
    RP#: 0, PG-Name: H
    RP#: 0, PG-Name: R
    RP#: 1, PG-Name: M1
    RP#: 1, PG-Name: H1
    RP#: 1, PG-Name: L1
    RP#: 1, PG-Name: R1
    RP#: 2, PG-Name: PGWL5
    RP#: 2, PG-Name: PGWL10
    PG#: 0, PG-Name:
    .
    .
    .
```

### Sample Output - Using PERFGROUPS to Show Priority Scheduler Settings Created in Teradata Viewpoint Workload Designer

This example shows a representative set of Priority Scheduler settings created in the Teradata Viewpoint Workload Designer portlet that are used by the system when TASM Workloads are enabled.

```
Submitting request TDWM PERFGROUPS;
Resource Partition items: 5
    Resource Partition Name : DEFAULT
    Resource Partition Name : TACTICAL
    Resource Partition Name : STANDARD
    Resource Partition Name :
    Resource Partition Name :
Performance Group items: 40
    RP#: 0, PG-Name: L
    RP#: 0, PG-Name: M
    RP#: 0, PG-Name: H
    RP#: 0, PG-Name: R
    RP#: 2, PG-Name: PGWL3
    RP#: 2, PG-Name: PGWL5
    RP#: 2, PG-Name: PGWL4
```

```

RP#: 2,  PG-Name: PGWL2
RP#: 2,  PG-Name: PGWL1
RP#: 0,  PG-Name: UNUSED9
RP#: 0.  PG-Name:
      .
      .
      .

```

## TDWM DELAY REQUEST CHANGE

Releases or aborts a specific request or session on the Teradata dynamic workload management software defer or delay queue.

### Input Data

| Element               | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte               | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode. |
| <i>mon_ver_id</i>     | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |
| <i>Request Flag</i>   | SMALLINT             | Indicator that the request or session was aborted or released: <ul style="list-style-type: none"> <li>• 0 = Request is aborted</li> <li>• 1 = Request is released</li> <li>• 2 = Session is aborted</li> <li>• 3 = Session is released</li> </ul>                                                                                                                                                                                                             |
| <i>Host ID</i>        | SMALLINT             | ID of the host number upon which the session containing the delayed request was established.                                                                                                                                                                                                                                                                                                                                                                  |
| <i>Reserved</i>       | SMALLINT             | Mod-4 alignment padding.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <i>Session Number</i> | INTEGER              | Number of the session in which the delayed request was submitted. A unique session number is assigned by the host (or client) at login.                                                                                                                                                                                                                                                                                                                       |
| <i>Request Number</i> | INTEGER              | Active request number.<br><br><b>Note:</b><br>This field is not used if <i>Request Flag</i> is either 2 or 3.                                                                                                                                                                                                                                                                                                                                                 |

| Element                  | Data Type | Description                                                                                                                                                                                                                                            |
|--------------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Workload Class Id</i> | INTEGER   | <p>Workload ID for the delayed request.</p> <p>If the request is from the context delay queue (where there is no workload ID), the field value is zero.</p> <p><b>Note:</b></p> <p>This field is not used if <i>Request Flag</i> is either 2 or 3.</p> |

## Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

The TDWM DELAY REQUEST CHANGE request allows requests or sessions to be processed out of the normal first-in-first-out (FIFO) order.

### Note:

Before you submit a TDWM DELAY REQUEST CHANGE request you need to run a TDWM STATISTICS or MONITOR SESSION request to obtain information about the deferred or delayed query that requires change. Use the output of the request to identify the *Host Id*, *Session Number*, *Request Number*, and *Workload Class Id* parameters required as part of the TDWM DELAY REQUEST CHANGE input.

If the identified request or session is not found in the Teradata dynamic workload management software delay queue, an error is returned. This condition is expected since requests or sessions are normally released from the delay queue whereas the information in the TDWM STATISTICS request could be outdated.

## CLv2 Response Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                                                            |
|-----------------|---------------|----------------|--------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273      | <p>StatementNo=1</p> <p>ActivityCount=0</p> <p>ActivityType=155 (PCLTWMDLREQSTCHGSTMT)</p> |

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields |
|-----------------|---------------|----------------|---------------------------------|
| EndStatement    | 11            | 6              | StatementNo = 2-byte integer=1  |
| EndRequest      | 12            | 4              | None                            |

### Sample Input - CLlv2 Request

The following example shows how the parcels for a TDWM DELAY REQUEST CHANGE request, built by CLlv2, appear when sent to the database server.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body                                                                                         |                                |
|--------|------|--------|----------------------------------------------------------------------------------------------|--------------------------------|
| Num    | Name | Bytes  | Field                                                                                        | Value                          |
| 0001   | Req  | 16     | Request                                                                                      | TDWM DELAY REQUEST CHANGE      |
| 0003   | Data | 24     | MonVerID<br>Request Flag<br>Host Id<br>Request Number<br>Session Number<br>Workload Class Id | 7<br>0<br>1<br>5<br>1000<br>10 |
| 0004   | Resp | 6      | BufferSize                                                                                   | 64000                          |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

The TDWM DELAY REQUEST CHANGE request returns values approximately as shown below when TASM Workloads are enabled and the following input data is specified:

- *Request Flag* = 0
- *Host Id* = 1
- *Request Number* = 5
- *Session Number* = 1000
- *Workload Class Id* = 10

The TDWM DELAY REQUEST CHANGE request commonly returns values in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
  StatementNo: 1    ActivityCount: 1
  ActivityType: 155 FieldCount: 1
DataInfo parcel:
  FieldCount: 1
EndStatement.
EndRequest.
```

## TDWM EXCEPTIONS

Collects Teradata dynamic workload management software exception data from the database.

### Input Data

| Element           | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte           | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data.</p> <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This must be version 8 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                                                                                                                                                                                               |

### Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

This request allows an application to receive the Teradata dynamic workload management software exception information from the database. There is one record per exception.

## CLv2 Response Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                        |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = PCLTWMEXPSTMT 199                                                                               |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                      |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100(record mode)</li> <li>6 to 64100(indicator mode)</li> </ul>   | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This is the only record returned                            |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                           |
| Success         | 8             | 18 to 273                                                                                                       | Statement 2<br>ActivityCount = number of record parcels returned<br>ActivityType = PCLTDWMEXPSTMT 199                                                  |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                      |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>8 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. There is one Exception record returned for each Exceptions. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                           |
| End Request     | 12            | 4                                                                                                               | None                                                                                                                                                   |

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

**Statement 1**

The first statement is a Record parcel format containing the collection duration data that is generated once for the whole system. The following table describes this Record format.

| Field/Column Name | Data Type | Description                                                                                                                |
|-------------------|-----------|----------------------------------------------------------------------------------------------------------------------------|
| SampleSec         | SMALLINT  | Duration of the collection period, in seconds. This value represents the collection rate as entered in Teradata Viewpoint. |

**Statement 2**

The second statement is a Record parcel format containing exception information for queries collected (and stored in a buffer) for the last collection period. There is one record for each exception. The following table describes this Record format.

| Field/Column Name  | Data Type                                 | Description                                                                       |
|--------------------|-------------------------------------------|-----------------------------------------------------------------------------------|
| VprocID            | SMALLINT                                  | Vproc the request with the exception was running on.                              |
| Query ID           | BIGINT                                    | Query ID for the query that encountered an exception.                             |
| UserName           | VARCHAR (128)<br>CHARACTER<br>SET UNICODE | User name used in the query with the exception.                                   |
| SessionID          | INTEGER                                   | Logical host ID of the query with the exception                                   |
| RequestNum         | SMALLINT                                  | Request number of the query with the exception                                    |
| LogicalHostID      | SMALLINT                                  | Logical host ID of the query with the exception.                                  |
| AcctString         | VARCHAR (128)<br>CHARACTER<br>SET UNICODE | Account string of the query with the exception.                                   |
| WDID               | INTEGER                                   | WD ID the query with the exception was running in.                                |
| OpEnvID            | INTEGER                                   | Planned environment ID in force when the query encountered the exception.         |
| SysConID           | INTEGER                                   | Health condition ID in force when the query encountered the exception.            |
| ClassificationTime | FLOAT                                     | Time the query with the exception was classified.                                 |
| ClassificationDate | INTEGER                                   | Date the query with the exception was classified.                                 |
| ExceptionTime      | FLOAT                                     | Time the query encountered the exception.                                         |
| ExceptionDate      | INTEGER                                   | Date the query encountered the exception.                                         |
| ExceptionValue     | INTEGER                                   | Type of exception that occurred:<br>• 0x00000001 - Exception time limit exceeded. |



| Field/Column Name | Data Type                                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |                                           | <ul style="list-style-type: none"> <li>• 0x00000002 - CPU time (AMP and PE) limit exceeded.</li> <li>• 0x00000004 - Blocked time limit exceeded.</li> <li>• 0x00000008 - Disk to CPU ratio exceeded.</li> <li>• 0x00000010 - AMP CPU skew limit exceeded.</li> <li>• 0x00000020 - AMP I/O count limit exceeded.</li> <li>• 0x00000040 - AMP I/O skew limit exceeded.</li> <li>• 0x00000080 - Max row count (for a step) exceeded.</li> <li>• 0x00000100 - Max row count (for a query) exceeded.</li> <li>• 0x00000200 - Spool space limit exceeded.</li> <li>• 0x00000400 - Number of AMPs used in query exceeded.</li> <li>• 0x00000800 - Disk CPU ratio value exceeded</li> <li>• 0x00001000 - I/O space value exceeded.</li> </ul> <p><b>Note:</b><br/>A conversion to hex is used to extract the bit values. For example, a value of 1024 converted to hex is 400.</p> |
| ExceptionAction   | VARCHAR (10)<br>CHARACTER<br>SET LATIN    | <p>Exception action taken by the exception handler:</p> <ul style="list-style-type: none"> <li>• A = Abort</li> <li>• C = Change WD<br/>NewWDId contains the new WD.</li> <li>• L = Log.</li> <li>• E = Execute Program<br/>ExProgram contains the program name.</li> <li>• T = Alert<br/>ExAlert contains the alert name.</li> <li>• N = No action<br/>This option cannot be combined with other actions. It disables exception detection.</li> <li>• S = Abort if the statement is a SELECT and no update has been done in the current transaction</li> <li>• Q = Insert a row to DBC.SystemQTbl</li> </ul> <p>For example, CE stands for change WD and execute program.</p>                                                                                                                                                                                             |
| NewWDID           | INTEGER                                   | WD the query was moved into if the ExceptionAction was change WD.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ExceptionCode     | INTEGER                                   | Database exception code.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| ExceptionSubCode  | INTEGER                                   | <p>Code for additional information.</p> <p><b>Note:</b><br/>This field is not currently used.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ErrorText         | VARCHAR (255)<br>CHARACTER<br>SET UNICODE | Error text generated for the exception.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| Field/Column Name | Data Type                                 | Description                                                                                              |
|-------------------|-------------------------------------------|----------------------------------------------------------------------------------------------------------|
| ExtraInfo         | VARCHAR (200)<br>CHARACTER<br>SET UNICODE | Exception values noted at the time of the exception.                                                     |
| RuleID            | INTEGER                                   | Teradata dynamic workload management software rule ID of the rule with the exception handling directive. |
| WarningOnly       | VARCHAR (1)<br>CHARACTER<br>SET LATIN     | Indicator that this is s a warning only.                                                                 |
| RejectionCat      | SMALLINT                                  | Teradata dynamic workload management software category this query was rejected from.                     |

### Sample Input - CLIV2 Request

The following example shows how the parcels for a TDWM EXCEPTIONS request, built by CLIV2, appear when sent to the database server.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body        |                 |
|--------|------|--------|-------------|-----------------|
| Num    | Name | Bytes  | Field       | Value           |
| 0001   | Req  | 20     | Request     | TDWM EXCEPTIONS |
| 0003   | Data | 6      | MonVerID    | 8               |
| 0004   | Resp | 6      | Buffer Size | 64000           |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

```
Submitting request TDWM EXCEPTIONS;
TDWM EXCEPTION successful.
    Sample Seconds: 60
TDWM Exceptions - # items: 3
User Name
Req Cat WIDID ExcptDate  ExcptTime  Code  NewWD  QueryID  ErrorText
```

```

-----
-----
LUMBER      2      0  11    2009/09/28  11:05:43.00    3156                16383718517360
1208  CPUTime: 687ms.
LUMBER      2      0  11    2009/09/28  11:05:57.00    3156                16383718517360
1276  CPUTime: 1501ms.
LUMBER      2      0  11    2009/09/28  11:05:57.00    3156                16383718517360
1276  I/O: 1720.

```

## TDWM LIST WD

Returns the names of the workloads.

### Input Data

| Element             | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte             | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode. |
| <i>mon_ver_id</i>   | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                   |
| <i>enabled_only</i> | SMALLINT             | Enabled workload names.<br><br><b>Note:</b><br>The values, 0 and 1, return enabled workload names only.                                                                                                                                                                                                                                                                                                                                                       |

### Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

The TDWM LIST WD request returns the WD information from the internal cache being used by the Teradata dynamic workload management software. TASM Workloads must be enabled through the Teradata Viewpoint Workload Designer portlet to use this request. WDs are returned in the order defined for evaluation.

## Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                 | Comments/Key Parcel Body Fields                                                                                                                                                                                           |
|-----------------|---------------|----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                      | StatementNo = 1<br>ActivityCount = 1 record for each WD being used at the time. There can a maximum of 246.<br>ActivityType = 161 (PCLTWMLISTWDSTMT)                                                                      |
| DataInfo        | 71            | 6 to 64100                                                                                                     | Optional: This parcel is present if request was IndicData parcel.                                                                                                                                                         |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100(indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is returned in record or indicator mode. There is one WD Information record returned for each WD satisfying the request. The format of this record is described below. |
| EndStatement    | 11            | 6                                                                                                              | StatementNo = 1                                                                                                                                                                                                           |
| EndRequest      | 12            | 4                                                                                                              | None                                                                                                                                                                                                                      |

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The following table describes the format of the WD Information Record parcel.

| Field/Column Name | Data Type    | Description                                       |
|-------------------|--------------|---------------------------------------------------|
| Configuration ID  | SMALLINT     | Configuration ID of the current rule set.         |
| Enabled Flag      | SMALLINT     | Indicator that the workload is currently enabled. |
| WLC ID            | INTEGER      | WD ID for the specified request.                  |
| WLC Name          | VARCHAR (30) | WD name.                                          |

| Field/Column Name | Data Type | Description                                                           |
|-------------------|-----------|-----------------------------------------------------------------------|
|                   |           | <b>Note:</b><br>Characters after the name length are not initialized. |

### Sample Input - CLlv2 Request

The following example shows how the parcels for a TDWM LIST WD request, built by CLlv2, appear when sent to the database server.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body                     |              |
|--------|------|--------|--------------------------|--------------|
| Num    | Name | Bytes  | Field                    | Value        |
| 0001   | Req  | 16     | Request                  | TDWM LIST WD |
| 0003   | Data | 4      | MonVerID<br>enabled_only | 7<br>1       |
| 0004   | Resp | 6      | BufferSize               | 64000        |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Sample Output

With an *enabled\_only* value of 1, this example shows only the enabled WDs on the current system in character text format for the LIST WD request when TASM Workloads are enabled. Your application program may return these values in a different format.

```
Submitting request TDWM LIST WD;
TDWM LIST WD successful.
```

```
Config ID    : 1
Enable Flag  : 1
WLC ID      : 3
Name Len    : 11
WLC Name    : WD-ConsoleH
```

```

Config ID   : 1
Enable Flag : 1
WLC ID      : 5
Name Len    : 11
WLC Name    : WD-ConsoleL
Config ID   : 1
Enable Flag : 1
WLC ID      : 4
Name Len    : 11
WLC Name    : WD-ConsoleM
Config ID   : 1
Enable Flag : 1
WLC ID      : 2
Name Len    : 11
WLC Name    : WD-ConsoleR
Config ID   : 1
Enable Flag : 1
WLC ID      : 1
Name Len    : 10
WLC Name    : WD-Default

```

## TDWM STATISTICS

Returns statistics from the Teradata dynamic workload management software.

### Input Data

| Element             | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndByte             | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data.</p> <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i>   | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This can be version 6 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                                                                                                                                                                                                |
| <i>Request Flag</i> | SMALLINT             | <p>Type of statistics returned:</p> <ul style="list-style-type: none"> <li>0 = System query counts</li> <li>1 = System session counts</li> </ul>                                                                                                                                                                                                                                                                                                                                     |

| Element | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|         |           | <ul style="list-style-type: none"> <li>• 2 = System delayed queries</li> <li>• 3 = Workload query counts</li> <li>• 4 = Workload delayed queries</li> </ul> <p>This request includes requests that are delayed due to WD throttle limits but no longer returns system delay queue information.</p> <ul style="list-style-type: none"> <li>• 5= (Load and archive) utility counts</li> <li>• 6 = Delayed utility session information</li> <li>• 7 = All delay statistics</li> </ul> <p>If you are using monitor version 13 or earlier, <i>Request Flag</i> requests 2, 4, and 6 are returned.</p> <p>If you are using monitor version 14 or later, <i>Request Flag</i> requests 2, 4, 6, and 16 are returned.</p> <ul style="list-style-type: none"> <li>• 8 = All information</li> </ul> <p>If you are using monitor software version 9 or earlier, <i>Request Flag</i> requests 1 through 7 are returned.</p> <p>If you are using monitor software version 10 through 13, <i>Request Flag</i> requests 1 through 7 and 9 are returned.</p> <p>If you are using monitor version 14 or later, <i>Request Flag</i> requests 1 through 7, 9, and 15 through 17 are returned.</p> <ul style="list-style-type: none"> <li>• 9 = All active requests</li> <li>• 15 = Return current arrival rates. If you are using monitor version 14 or later, current rates of enabled arrival rate meters are returned.</li> <li>• 16 = Arrival rate meter deferred requests</li> </ul> <p>If you are using monitor version 14 or later, requests that are currently deferred due to one or more arrival rate meters are returned.</p> <ul style="list-style-type: none"> <li>• 17 = Return detail current arrival rates</li> </ul> <p>If you are using monitor version 14 or later, details about current rates of enabled arrival rate meters are returned. For every ARM, the time unit is divided into 6 intervals and the numbers of arriving requests for each interval are returned in Statement 5 records.</p> <p><b>Note:</b></p> <p><i>Request Flag:</i></p> <ul style="list-style-type: none"> <li>• 7 is valid for monitor version software ID 8 or later.</li> <li>• 8 is valid for monitor version software ID 9 or later.</li> <li>• 9 is valid for monitor version software ID 10 or later.</li> <li>• 15 through 17 are valid for monitor version software ID 14 or later.</li> </ul> |

## Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100

- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

Before using this request, see [Impact of Object Name Length on PM/API Requests](#).

The Workload Query Manager (WQM) enforces TASM rules on the number of active system sessions, system queries, workloads, workload groups, and load or archive utilities for specified conditions.

The TDWM STATISTICS request returns information in the various record formats depending on the *Request Flag* supplied as described in the following table.

The *Request Flag* in the input record determines the format of the data in the response records, and in some cases the type of information contained within each statement (for example, a query or session).

Use Teradata Viewpoint to learn more about the WQM process.

The following table specifies which format types are returned.

### Note:

These formats are defined later in this section.

| If a Request Flag is...          | Format1<br>Records<br>Are<br>Returned... | Format2<br>Records<br>Are<br>Returned... | Format3<br>Records<br>Are<br>Returned... | Format 4<br>Records<br>Are<br>Returned... | Format 5<br>Records<br>Are<br>Returned... |
|----------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|-------------------------------------------|-------------------------------------------|
| 0 = System query counts          | X                                        |                                          |                                          |                                           |                                           |
| 1 = System session counts        | X                                        |                                          |                                          |                                           |                                           |
| 2 = System<br>delayed requests   |                                          | X                                        |                                          |                                           |                                           |
| 3 = Workload query counts        | X                                        |                                          |                                          |                                           |                                           |
| 4 = Workload<br>delayed requests |                                          | X                                        |                                          |                                           |                                           |
| 5 = Utility counts               |                                          |                                          | X                                        |                                           |                                           |
| 6 = Utility delayed sessions     |                                          | X                                        |                                          |                                           |                                           |
| 7 = All delay information        |                                          | X                                        |                                          |                                           |                                           |
| 8 = All information              | X                                        | X                                        | X                                        | X                                         | X                                         |



| If a Request Flag is...                                                                                                                                                                                                                                                                                                                                                       | Format1<br>Records<br>Are<br>Returned... | Format2<br>Records<br>Are<br>Returned... | Format3<br>Records<br>Are<br>Returned... | Format 4<br>Records<br>Are<br>Returned... | Format 5<br>Records<br>Are<br>Returned... |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|------------------------------------------|------------------------------------------|-------------------------------------------|-------------------------------------------|
| <b>Note:</b><br>If you are using monitor software version 9 or earlier, <i>Request Flag</i> requests 1 through 7 are returned. If you are using monitor software version 10 or later, <i>Request Flag</i> requests 1 through 7 and 9 are returned. If you are using monitor version 14 or later, <i>Request Flag</i> requests 1 through 7, 9, and 15 through 17 are returned. |                                          |                                          |                                          |                                           |                                           |
| 9 = All active requests                                                                                                                                                                                                                                                                                                                                                       |                                          |                                          |                                          | X                                         |                                           |
| 15 = Arrival rate meter counts                                                                                                                                                                                                                                                                                                                                                | X                                        |                                          |                                          |                                           |                                           |
| 16 = Arrival rate meter deferred requests                                                                                                                                                                                                                                                                                                                                     |                                          | X                                        |                                          |                                           |                                           |
| 17 = Detail arrival rate meter counts                                                                                                                                                                                                                                                                                                                                         |                                          |                                          |                                          |                                           | X                                         |

## CLv2 Response Parcels

The following is the Parcel Sequence returned if *Request Flag* 0 through 7 is requested. Only one parcel is returned.

Statement 1, Statement 2, Statement 3, or Statement 4 is returned depending on the *Request Flag*. Since a system query, system session, or utility throttle can be affected by multiple rules, multiple rows can be returned for requests 2, 4, 6, 7, 8, and 9.

| Parcel Sequence | Parcel Flavor | Length (Bytes)             | Comments/Key Parcel Body Fields                                                                                                                |
|-----------------|---------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                  | StatementNo = 1, 2, or 3<br>ActivityCount = Not applicable<br>ActivityType = 132 (PCLTWMSTATSSTMT)                                             |
| DataInfo        | 71            | 6 to 64100                 | Optional; this parcel is present if request was IndicData parcel.                                                                              |
| Record          | 10            | • 5 to 64100 (record mode) | Depending on the request (Data or IndicData), data is returned in record or indicator mode. This record contains information in StatementNo-1, |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                | Comments/Key Parcel Body Fields                                                                                                                                                                                                |
|-----------------|---------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |               | <ul style="list-style-type: none"> <li>6 to 64100 (indicator mode)</li> </ul> | StatementNo-2, or StatementNo-3 data format, depending on the Request Flag in the input data. There is one Record parcel for each item being tracked by the WQM. If there are no qualifying records, no Record parcel is sent. |
| EndStatement    | 11            | 6                                                                             | StatementNo = 2-byte integer=1, 2, or 3                                                                                                                                                                                        |
| EndRequest      | 12            | 4                                                                             | None                                                                                                                                                                                                                           |

The following Parcel Sequence is returned if Request Flag is 8 or 9 and the monitor version software ID is 10 or greater.

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                                                                                                                |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 1<br>ActivityCount = number of queries with applicable statistics<br>ActivityType = 132 = PCLTWMSTATSSTMT                                                                                                                                                        |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                                                                                                                                              |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>7 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo-1. There is one record parcel for each item being tracked by the WQM. If there are no qualifying records, no record parcel is sent. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer = 1                                                                                                                                                                                                                                               |
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 2<br>ActivityCount = number of queries with applicable statistics<br>ActivityType = 132 = PCLTWMSTATSSTMT                                                                                                                                                        |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                                                                                                                                              |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100(record mode)</li> <li>7 to 64100 (indicator mode)</li> </ul>  | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo 2. There is one record parcel for each item being tracked by the WQM. If there are no qualifying records, no record parcel is sent. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer = 2                                                                                                                                                                                                                                               |
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 3                                                                                                                                                                                                                                                                |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                                                                                                                |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 |               |                                                                                                                 | ActivityCount = number of queries with applicable statistics<br>ActivityType = 132 = PCLTWMSTATSSTMT                                                                                                                                                                           |
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                                                                                                                                                                              |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>7 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo 3. There is one record parcel for each item being tracked by the WQM. If there are no qualifying records, no record parcel is sent. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer = 3                                                                                                                                                                                                                                               |
| Success         | 8             | 18 to 273                                                                                                       | StatementNo = 4-byte<br>ActivityCount = number of queries with applicable statistics<br>ActivityType = 132 = PCLTWMSTATSSTMT                                                                                                                                                   |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>7 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This record contains information in StatementNo 4. There is one record parcel for each item being tracked by the WQM. If there are no qualifying records, no record parcel is sent. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer = 4                                                                                                                                                                                                                                               |
| EndRequest      | 12            | 4                                                                                                               | None                                                                                                                                                                                                                                                                           |

### Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

### Response

#### Note:

Each of the statement types described below correspond to a `ResultSet` returned by the Teradata JDBC Driver, and each statement type field corresponds to a `ResultSet` column. For more information on `ResultSet`s, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

**Statement 1**

The first statement is a Record parcel format containing statistical information for one system query, system session, or workload item. There is one record for each item that is being tracked by the Teradata dynamic workload management software. The following table describes this Record format.

| Field/Column Name | Data Type                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Record Type       | SMALLINT                               | Type of information contained in this record: <ul style="list-style-type: none"> <li>• 1 = User</li> <li>• 2 = Account String</li> <li>• 4 = Account String</li> <li>• 5 = Profile</li> <li>• 6 = Client ID</li> <li>• 7 = Network Address</li> <li>• 8 = Application</li> <li>• 9 = Collective Rule</li> <li>• 10 = Database</li> <li>• 11 = Table</li> <li>• 12 = WD</li> <li>• 13 = Utility</li> <li>• 14 = Macro</li> <li>• 15 = Stored Procedure</li> <li>• 16 = View</li> <li>• 17 = QueryBand</li> <li>• 18 = Function</li> <li>• 19 = Method</li> <li>• 20 = Virtual partition</li> <li>• 21 = Reserved</li> <li>• 22 = Server</li> <li>• 23 = AWT Resource Limit</li> <li>• 24 = Arrival Rate Meter</li> </ul> |
| Rule ID           | INTEGER                                | Rule ID for the system query, system session, or workload for this statistic.<br>If <i>Request Flag</i> is 0 or 1, Rule ID returns the ID of the system query or system session throttle rule.<br>If the <i>Request Flag</i> is 3, Rule ID returns the ID of the workload.<br>If Record Type is 24, Rule ID returns the ID of the arrival rate meter.                                                                                                                                                                                                                                                                                                                                                                   |
| Object Name       | VARCHAR (128)<br>CHARACTER SET UNICODE | For <i>Request Flags</i> 0 and 1, this is the name of the system query or system session throttle rule based on its Record Type. When Record Type is set to 11, 14, 15, 16, 18 or 19, this field is the qualifying database name for the <i>Table Name</i> field.<br>For <i>Request flag</i> 3, this field is always empty.                                                                                                                                                                                                                                                                                                                                                                                             |

| Field/Column Name | Data Type                              | Description                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Qualified Name    | VARCHAR (128)<br>CHARACTER SET UNICODE | For <i>Request Flags</i> 0 and 1, this is the name of the table, macro, stored procedure, function or method object when Record Type is set to 11, 14, 15, 16, 18, or 19.<br>For <i>Request Flag</i> 3, this field is always empty.                                                                                                                                                            |
| Active            | INTEGER                                | Number of requests currently active. This is the number of active queries or sessions depending on the information requested in the input record.<br>If Record Type is 24, Active returns the current arrival rate for the time unit specified in the TimeUnit field.                                                                                                                          |
| Limit             | INTEGER                                | Current minimum limit on this item. This is the limit on the number of queries or sessions depending on the information requested in the input record.<br>If Record Type is 24, Limit returns the current arrival rate limit for the time unit specified in the TimeUnit field. A value of 1 indicates that no limit is defined.                                                               |
| Delayed           | INTEGER                                | Number of requests currently delayed (queued) for this item.<br><b>Note:</b><br>This field is not applicable if <i>Request Flag</i> is 1.<br>If Record Type is 24, Delayed returns the number of requests currently in the Defer queue for this arrival rate meter. This value is not meaningful when Limit is -1.                                                                             |
| Rule Name         | VARCHAR (30)<br>CHARACTER SET UNICODE  | Rule name for the system query or system session throttle rule, or workload for this statistic.<br>If <i>Request Flag</i> is 0 or 1, Rule Name returns the name of the system query or system session throttle rule is returned.<br>If the <i>Request Flag</i> is 3, Rule Name returns the name of the workload.<br>If Record Type is 24, RuleName returns the name of the arrival rate meter. |
| Statistic Type    | INTEGER                                | Type of statistics returned: <ul style="list-style-type: none"> <li>• 0 = Query data</li> <li>• 1 = Logon data</li> <li>• 2 = Workloads</li> </ul> <b>Note:</b><br>This field is only valid on monitor version software ID 8 or later.                                                                                                                                                         |
| FlexEligible      | SMALLINT                               | The delayed request that is in a workload and is tagged as eligible for the Flex Throttle feature. Note, that flagged requests must meet all requirements for Flex Throttles before being released, such as system throttle restrictions. <ul style="list-style-type: none"> <li>• 0 = Flex eligible</li> <li>• 1 = Not Flex eligible</li> </ul>                                               |
| TimeUnit          | SMALLINT                               | If Record Type is 24, this field is the time unit of this arrival rate meter:                                                                                                                                                                                                                                                                                                                  |

| Field/Column Name           | Data Type | Description                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |           | <ul style="list-style-type: none"> <li>• 1 = second</li> <li>• 2 = minute</li> <li>• 3 = hour</li> </ul> If Record Type is not 24, this field is not meaningful.                                                                                                                                                                                          |
| Required Qualification Time | INTEGER   | If Record Type is 24, Required Qualification Time returns the number of seconds in which the current arrival rate must continuously remain at the limit before new requests are held in the Defer queue. A value of zero indicates the limit will be enforced as soon as it is reached.           If Record Type is not 24, this field is not meaningful. |
| Current Qualification Time  | INTEGER   | If Record Type is 24, Current Qualification Time returns the number of seconds in which the current arrival rate has continuously remained at the limit. A value of zero indicates the limit has not been reached.           If Record Type is not 24, this field is not meaningful.                                                                      |

### Sample Output - Statement 1: Using TDWM STATISTICS with Request Flag Set to 0

If a *Request Flag* of zero (system query counts) is requested, the following is an example of the records that are returned as part of the first statement.

| RecordType | Rule ID | Obj Name | Qual Name | Active | Limit | Delayed |
|------------|---------|----------|-----------|--------|-------|---------|
| -----      | -----   | -----    | -----     | -----  | ----- | -----   |
| 1          | 0       | Ken      | 4         | 5      | 0     |         |
| 1          | 0       | Dina     | 2         | 4      | 0     |         |
| 2          | 0       | Sales    | 3         | 10     | 0     |         |
| 2          | 0       | PreSales | 2         | 10     | 0     |         |
| 2          | 0       | Training | 3         | 3      | 10    |         |

This example shows the following important information:

- Two users running (Record Type 1) also have rules defined on the number of queries that they are allowed to have running simultaneously (Limit).
- Three accounts running (Record Type 2) and also have rules defined on the number of queries that they are allowed to have running simultaneously (Limit).

Since the input requested query information (a *Request Flag* of zero), the numbers shown reflect the number of active *queries* for each user and account. If logon information were requested (a *Request Flag* of 1) in the input, the records would look the same. However, the numbers shown would reflect the number of active sessions within the restricted area.

Items that are restricted, but are not currently active on the system, or are bypassed, are not be returned.

**Sample Output - Statement 1: Using TDWM STATISTICS with Tables Restricted (RecordType 11)**

If Tables are restricted, the following records are returned:

| RecordType | Rule ID | Obj Name | Qual Name  | Active Name | Limit | Delayed |
|------------|---------|----------|------------|-------------|-------|---------|
| -----      | -----   | -----    | -----      | -----       | ----- | -----   |
| 11         | 0       | Employee | PayrollDB  | 2           | 10    | 0       |
| 11         | 0       | Leads    | MaketingDB | 1           | 1     | 1       |

**Sample Output - Statement 1: Using TDWM STATISTICS with Request Flag Set to 3**

If WDs (Request Flag 3) are requested, the following records are returned:

| RecordType | Rule ID | Obj Name    | Qual Name | Active Name | Limit | Delayed |
|------------|---------|-------------|-----------|-------------|-------|---------|
| -----      | -----   | -----       | -----     | -----       | ----- | -----   |
| 12         | 03      | TacticalWD  |           | 4           | 100   | 0       |
| 12         | 05      | StrategicWD |           | 2           | 2     | 1       |
| 12         | 05      | AdHocWD     |           | 0           | -1    | 0       |

**Note:**

-1 limit means that there is no limit.

**Sample Output - Statement 1: Using TDWM STATISTICS with Request Flag Set to 8**

If Request Flag 8 is requested, the following records are returned:

| RecordType | Rule ID   | Obj Name    | Qual Name  | Active | Limit | Delayed | Rule          |
|------------|-----------|-------------|------------|--------|-------|---------|---------------|
| Name       | Stat Type |             |            |        |       |         |               |
| -----      | -----     | -----       | -----      | -----  | ----- | -----   |               |
| -----      | -----     |             |            |        |       |         |               |
| 0          | 0         | Ken         |            | 4      | 5     | 0       |               |
| 0          | 0         | Dina        |            | 2      | 4     | 0       |               |
| 1          | 0         | Sales       |            | 3      | 10    | 0       |               |
| 1          | 0         | Presales    |            | 2      | 10    | 0       |               |
| 1          | 0         | Training    |            | 3      | 3     | 10      | LimitTraining |
| 2          | 0         | H           |            | 6      | 6     | 2       |               |
| 2          | 0         | H2          |            | 1      | 1     | 0       |               |
| 11         | 0         | Employee    | PayrollDB  | 2      | 10    | 0       |               |
| 11         | 0         | Leads       | MaketingDB | 1      | 1     | 1       |               |
| 12         | 3         | TacticalWD  |            | 44     | 100   | 0       |               |
| 12         | 5         | StrategicWD |            | 2      | 2     | 1       |               |

|    |     |         |    |    |   |       |
|----|-----|---------|----|----|---|-------|
| 12 | 5   | AdHocWD | 0  | -1 | 0 |       |
| 24 | 117 |         | 10 | 10 | 5 | ARM_1 |

## Statement 2

The second statement is a Record parcel format containing information about queries or sessions being delayed by the Teradata dynamic workload management software due to a system query, system session, workload or utility throttle limit. If Request Flag 7 is requested, one record for each request or session that is being delayed is returned with one Rule ID. Multiple rows are returned if the request has more than one rule causing its delay.

If Request Flag 16 is requested, the record parcel reports similar information about queries being deferred due to one or more arrival rate meters. If a request is deferred due to multiple arrival rate meters, one row is returned for each rule causing its deferral.

The following table describes this Record format.

| Field/Column Name | Data Type                              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User name         | VARCHAR (128)<br>CHARACTER SET UNICODE | Logon user name of session.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| HostId            | SMALLINT                               | Host ID of the session number for the delayed request or session.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Session Number    | INTEGER                                | Number for the held request or session.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Request Number    | INTEGER                                | Request number for the delayed request or session.<br>This field is zero if <i>Request Flag</i> is 6.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Rule Name         | VARCHAR (30)<br>CHARACTER SET UNICODE  | Rule name for the system query or session throttle rule, workload, or arrival rate meter rule for this statistic.<br>If <i>Request Flag</i> is 2 or 6, Rule Name is the name of the system query or system session throttle rule.<br>If <i>Request Flag</i> is 4, Rule Name returns the name of the workload.<br>If <i>Request Flag</i> is 9, Rule Name and all remaining fields are empty, except for Rule Type.<br>If <i>Request Flag</i> is 16, Rule Name is the name of the arrival rate meter rule.                     |
| Rule ID           | INTEGER                                | Rule ID for the system query or system session throttle, workload, or arrival rate meter for this statistic.<br>If <i>Request Flag</i> is 2 or 6, Rule ID returns the ID of the system query or system session throttle rule.<br>If <i>Request Flag</i> is 4, Rule ID returns the ID of the workload.<br>This Rule ID must be supplied in the Abort/Release interface (when used). If the <i>Request Flag</i> is 6, Rule ID is zero.<br>If <i>Request Flag</i> is 16, Rule ID returns the ID of the arrival rate meter rule. |



| Field/Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Total Time Held   | INTEGER   | Total number of wall clock seconds that this request or session has been held by the WQM.<br>If Rule Type is 6, returns the total number of wall clock seconds that this request has been held in the Defer queue due to one or more arrival rate meters.                                                                                                                                                                                                                    |
| Overridable       | Boolean   | Indicator that the database administrator is allowed to abort or release the request.<br><b>Note:</b><br>The queue table requests are controlled internally by the database and cannot be altered by the administrator.<br>If <i>Request Flag</i> is 6, this field indicates if the delayed session can be released. A session cannot be released if it exceeds the internal AMP worker task limit or an internal utility limit.<br>A delayed session can always be aborted. |
| Blocking Count    | INTEGER   | Number of consecutive times that this session has been identified as blocking at least one other session.<br>If <i>Request Flag</i> is 6, Blocking Count is zero.                                                                                                                                                                                                                                                                                                            |
| IFP ID            | SMALLINT  | ID of the PE vproc that initiated this request.                                                                                                                                                                                                                                                                                                                                                                                                                              |
| WD Delayed        | INTEGER   | <ul style="list-style-type: none"> <li>• 1 = Query is delayed for a workload throttle rule.</li> <li>• 0 = Query is not delayed for a workload throttle rule.</li> </ul> <b>Note:</b><br>This field is only valid on monitor version software ID 8 or later.                                                                                                                                                                                                                 |
| System Delayed    | INTEGER   | <ul style="list-style-type: none"> <li>• 1 = Query is delayed for a system query or system session throttle rule.</li> <li>• 0 = Query is not delayed for a system query or system session throttle rule.</li> </ul> <b>Note:</b><br>This field is only valid on monitor version software ID 8 or later.                                                                                                                                                                     |
| Utility Delayed   | INTEGER   | <ul style="list-style-type: none"> <li>• 1 = Query is delayed for a utility throttle rule.</li> <li>• 0 = Query is not delayed for a utility throttle rule.</li> </ul> <b>Note:</b><br>This field is only valid on monitor version software ID 8 or later.                                                                                                                                                                                                                   |
| Rule Type         | INTEGER   | Rule type for the request: <ul style="list-style-type: none"> <li>• 0 = Workload</li> <li>• 1 = System</li> <li>• 2 = Utility</li> <li>• 3 = Workload Group *</li> <li>• 4 = Virtual Partition *</li> </ul>                                                                                                                                                                                                                                                                  |

| Field/Column Name           | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |           | <ul style="list-style-type: none"> <li>• 5 = AWT Resource Limit **</li> <li>• 6 = Arrival Rate Meter ***</li> </ul> <p><b>Note:</b><br/>The Rule Type field is only valid on monitor version software ID 8 or later.<br/>* Workload Group 3 and Virtual Partition 4 are returned in Monitor Version 11 and later.<br/>** AWT Resource Limit is returned in Monitor Version 12 and later.<br/>*** Arrival Rate Meter is returned in monitor version 14 and later.</p> |
| Group Delayed               | INTEGER   | <ul style="list-style-type: none"> <li>• 1= Query is delayed for a workload group throttle.</li> <li>• 0= Query is not delayed for a workload group throttle.</li> </ul> <p><b>Note:</b><br/>This field is only valid on monitor version software ID 10 or later.</p>                                                                                                                                                                                                |
| FlexEligible                | SMALLINT  | <p>The delayed request that is in a workload and is tagged as eligible for the Flex Throttle feature. Note, that flagged requests must meet all requirements for Flex Throttles before being released, such as system throttle restrictions.</p> <ul style="list-style-type: none"> <li>• 0 = Flex eligible</li> <li>• 1 = Not Flex eligible</li> </ul>                                                                                                              |
| Arrival Rate Meter Deferred | INTEGER   | <ul style="list-style-type: none"> <li>• 1 = Query is deferred due to an arrival rate meter rule.</li> <li>• 0 = Query is not deferred due to an arrival rate meter rule.</li> </ul>                                                                                                                                                                                                                                                                                 |

### Sample Output - Statement 2: Using TDWM STATISTICS with Request Flag Set to 6

If a *Request Flag* of 6 is requested, the following Records are returned as part of the second statement:

| User Name | Host ID | Sess | Req Nbr | Rule Name | Rule ID | Total TimeHeld | Over | Block |
|-----------|---------|------|---------|-----------|---------|----------------|------|-------|
| -----     | -----   | ---- | -----   | -----     | -----   | -----          | ---- | ----- |
| Ken       | 01      | 1012 | 0       |           | 0       | 45             | True | 0     |
| Dina      | 01      | 1088 | 0       |           | 0       | 110            | True | 0     |

This report indicates that 2 sessions are being delayed due to a system query or system session throttle limits.

### Sample Output - Statement 2: Using TDWM STATISTICS with Request Flag Set to 7

If a *Request Flag* of 7 is requested, the following Records are returned as part of the second statement:

| User Name | Host ID | Sess | Req Nbr | Rule Name | Rule ID | Total TimeHeld | Over | Block |
|-----------|---------|------|---------|-----------|---------|----------------|------|-------|
| Ken       | 01      | 1013 | 1       |           | 4       | 15             | True | 0     |
| Ken       | 01      | 1013 | 1       |           | 5       | 15             | True | 0     |
| Dina      | 01      | 1089 | 0       |           | 4       | 110            | True | 0     |

This report indicates that 2 requests are being delayed, but the first request has two rules that are causing its delay.

### Sample Output - Statement 2: Using TDWM STATISTICS with Request Flag Set to 16

| User Name | Host ID | Sess | Req Nbr | Rule Name | Rule ID | Total TimeHeld | Over | Block |
|-----------|---------|------|---------|-----------|---------|----------------|------|-------|
| USER1     | 1       | 4734 | 2       | ARM_1     | 117     | 17             | True | 0     |

### Statement 3

The third statement is a Record parcel format containing information on the load utilities that are active in the system. There is one record for every load utility type.

This statement is only returned when utility statistics are requested. The following table describes this record format.

| Field/Column Name | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Utility Type      | SMALLINT  | Type of utility information contained in this record: <ul style="list-style-type: none"> <li>• 0 = MultiLoad + FastLoad</li> <li>• 1 = MultiLoad</li> <li>• 2 = FastLoad</li> <li>• 3 = FastExport</li> <li>• 4 = ARC</li> <li>• 5 = Standalone Mload</li> <li>• 6 = Standalone FastLoad</li> <li>• 7 = Standalone FastExport</li> <li>• 8 = TPT update Op: MultiLoad</li> <li>• 9 = TPT load Op: Fastload</li> <li>• 10 = TPT export Op: FastExport</li> <li>• 11 = JDBC MultiLoad</li> <li>• 12 = JDBC FastLoad</li> <li>• 13 = JDBC FastExport</li> <li>• 14 = CSP Save Dump (FastLoad)</li> <li>• 15 = DSA</li> <li>• 16 = DSA Backup</li> <li>• 17 = DSA Restore</li> <li>• 18 = MLOADX</li> </ul> |

| Field/Column Name | Data Type | Description                                                                                                  |
|-------------------|-----------|--------------------------------------------------------------------------------------------------------------|
| Count             | SMALLINT  | Number of utilities of this utility type that are active.                                                    |
| Limit             | SMALLINT  | Current limit on the number of utilities of this type.<br>A value of -1 indicates there is no limit defined. |

### Sample Output - Statement 3: Using TDWM STATISTICS

The following is an example of the records that are returned as part of third statement:

| TYPE | COUNT | LIMIT |
|------|-------|-------|
| ---- | ----- | ----- |
| 0    | 0     | 30    |
| 1    | 0     | 30    |
| 2    | 0     | 30    |
| 3    | 0     | 60    |
| 4    | 0     | 350   |
| 5    | 0     | 30    |
| 6    | 0     | 30    |
| 7    | 0     | 60    |
| 8    | 0     | 30    |
| 9    | 0     | 30    |
| 10   | 0     | 60    |
| 11   | 0     | 30    |
| 12   | 0     | 30    |
| 13   | 0     | 60    |
| 14   | 0     | 30    |

Since a row is returned for each utility type, the Limit value can be either the system default value for the utility or a TASM rule Throttle value.

#### Note:

When a TASM rule Throttle value is active, it overrides the default limit.

### Statement 4

The fourth statement is a Record parcel format containing information about queries, sessions, and so on being counted as active by the Teradata dynamic workload management software due to a system query, system session, workload, workload group, or utility throttle rule.

Multiple rows in this statement are returned if the request has more than one rule for which it is counted as active.

**Note:**

This statement is not a count of all active queries in the system.

| Field/Column Name       | Data Type                                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User name               | VARCHAR (128)<br>CHARACTER<br>SET UNICODE | Logon user name of session.                                                                                                                                                                                                                                                                                                                                                                                                                |
| HostId                  | SMALLINT                                  | Session number ID for the active request.                                                                                                                                                                                                                                                                                                                                                                                                  |
| Session Number          | INTEGER                                   | Session number for the active request.                                                                                                                                                                                                                                                                                                                                                                                                     |
| Request Number          | INTEGER                                   | Request number for the active request.                                                                                                                                                                                                                                                                                                                                                                                                     |
| Internal Request Number | INTEGER                                   | Internal AMP request number for the active request.                                                                                                                                                                                                                                                                                                                                                                                        |
| Rule Name               | VARCHAR(30)<br>CHARACTER<br>SET UNICODE   | Rule name for the active request.                                                                                                                                                                                                                                                                                                                                                                                                          |
| Rule Id                 | INTEGER                                   | Rule ID for the active request.                                                                                                                                                                                                                                                                                                                                                                                                            |
| Rule Type               | INTEGER                                   | Rule type for the request: <ul style="list-style-type: none"> <li>• 0 = Workload class</li> <li>• 1 = System query</li> <li>• 2 = System session</li> <li>• 3 = Utility throttle</li> <li>• 4 = Workload group</li> <li>• 5 = Virtual Partition *</li> <li>• 6 = AWT Resource Limit **</li> </ul> * Virtual Partition 5 is returned in Monitor Version 11 and later.<br>** AWT Resource Limit is returned in Monitor Version 12 and later. |
| Statement Type          | INTEGER                                   | Type of active request: <ul style="list-style-type: none"> <li>• 0 = Session</li> <li>• 1 = Query</li> <li>• 2 = Call</li> <li>• 3 = Unit of work</li> </ul>                                                                                                                                                                                                                                                                               |
| Time Active             | INTEGER                                   | Total number of wall clock seconds that this request has been active.                                                                                                                                                                                                                                                                                                                                                                      |
| ActiveThrottleBypassed  | SMALLINT                                  | Whether an active request is active solely due to the ThrottleBypass ruleset attribute. This attribute overrides the throttle limits if the session owning the request has an object lock higher than the Access level. Possible values include: <ul style="list-style-type: none"> <li>• 0 = This value indicates one of the following:<br/>The ThrottleBypass ruleset attribute is not active.</li> </ul>                                |

| Field/Column Name | Data Type | Description                                                                                                                                                                                                                                                                                   |
|-------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |           | <p>The throttle limit is lower than the Access level.<br/>The throttle limit is not specified.</p> <ul style="list-style-type: none"> <li>• 1 = The ThrottleBypass ruleset attribute is active.</li> </ul> <p>For more information, see <i>Teradata® Viewpoint User Guide</i>, B035-2206.</p> |
| FlexReleased      | SMALLINT  | <p>An active request running due to the Flex Throttle feature.</p> <ul style="list-style-type: none"> <li>• 0 = not Flex released</li> <li>• 1 = Flex released</li> </ul>                                                                                                                     |

### Sample Output - Statement 4: Using TDWM STATISTICS with Active Query Requests

The following example shows two active query requests. One of the active query requests is due to a TASM Workload rule and one is due to a TASM Utility throttle rule.

TDWM Stats - Active items: 2

| USERNAME | HOST | SESSION | REQ | INTREQ | RULEID / NAME | TYPE     | STMTTYPE | ACTIVETIME |
|----------|------|---------|-----|--------|---------------|----------|----------|------------|
| TEST1    | 1025 | 1000    | 6   | 6      | (15)WD-ALT    | WD       | 1        | 7          |
| TEST1    | 1025 | 1000    | 6   | 6      | (13)WD-GRP1   | Util/Ses | 1        | 7          |

### Sample Output - Statement 4: Using TDWM STATISTICS with Flex Throttle Query Requests

The following example shows active query requests. Some of the active query requests are running because the Flex Throttle feature released them to run.

Active Requests:

| HOST               | SESSION | INT | REQ | TYP | DEP | USER | NAME | RULE | TYPE (ID) | NAME         | ACTIVE | ... | FLEXRELD |
|--------------------|---------|-----|-----|-----|-----|------|------|------|-----------|--------------|--------|-----|----------|
| /QUERYID           |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22638   |     | 2   | QRY | 1   | KEN9 |      |      |           |              |        |     |          |
| WLC( 99)WD-9       |         |     | 349 |     |     |      |      |      |           |              |        |     |          |
| 307193492630017729 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22643   |     | 2   | QRY | 1   | KEN8 |      |      |           |              |        |     |          |
| WLC( 98)WD-8       |         |     | 346 |     |     |      |      |      |           |              |        |     |          |
| 307183492630017055 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22648   |     | 2   | QRY | 1   | KEN7 |      |      |           |              |        |     |          |
| WLC( 97)WD-7       |         |     | 345 |     |     |      |      |      |           |              |        |     |          |
| 307193492630017748 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22655   |     | 2   | QRY | 1   | KEN5 |      |      |           |              |        |     |          |
| WLC( 95)WD-5       |         |     | 340 |     |     |      |      |      |           |              |        |     |          |
| 307183492630017095 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22655   |     | 2   | QRY | 1   | KEN5 |      |      |           | WLG( 90)WD-  |        |     |          |
| GRP45              |         |     | 340 |     |     |      |      |      |           |              |        |     |          |
| 307183492630017095 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22663   |     | 2   | QRY | 1   | KEN3 |      |      |           |              |        |     |          |
| WLC( 93)WD-3       |         |     | 337 |     |     |      |      |      |           |              |        |     |          |
| 307183492630017124 |         |     |     |     |     |      |      |      |           |              |        |     |          |
| 1                  | 22664   |     | 2   | QRY | 1   | KEN3 |      |      |           | WLC( 93)WD-3 | 321    |     | YES      |

```

307193492630017784
 1      22647      2  QRY   1  KEN7      WLC( 97)WD-7      261      YES
307183492630017065
 1      22642      2  QRY   1  KEN8      WLC( 98)WD-8      201      YES
307193492630017738
 1      22644      2  QRY   1  KEN8      WLC( 98)WD-8      141      YES
307193492630017739
 1      22637      2  QRY   1  KEN9      WLC( 99)WD-9       81      YES
307183492630017034

```

## Statement 5

Statement 5 record format returns detailed information about each arrival rate meter. An arrival rate is represented as a request count per time unit. Each time unit is divided into six fixed intervals. Each row and record contains information for one interval. Therefore, there are six rows for each arrival rate meter.

| Field Name              | Data Type                          | Description                                                    |
|-------------------------|------------------------------------|----------------------------------------------------------------|
| Rule Type               | INTEGER                            | Rule type for the request:<br>• 0 = Arrival Rate Meter         |
| Rule Id                 | INTEGER                            | ID of the arrival rate meter.                                  |
| Rule Name               | VARCHAR (30)<br>CHARSET<br>UNICODE | Name of the arrival rate meter.                                |
| Interval Number         | INTEGER                            | Interval number.                                               |
| Interval Start Date     | INTEGER                            | Start date of this interval.                                   |
| Interval Start Time     | FLOAT                              | Start time of this interval.                                   |
| Request Count           | INTEGER                            | Number of requests admitted in this interval.                  |
| Host ID                 | SMALLINT                           | HOST ID of the first request in this interval.                 |
| Session Number          | INTEGER                            | Session number of the first request in this interval.          |
| Request Number          | INTEGER                            | Request number of the first request in this interval.          |
| Internal Request Number | INTEGER                            | Internal request number of the first request in this interval. |

## Sample Output - Statement 5

```

TDWM Stats - ARM detail count items: 12
TYPE RULE (ID) NAME      ITVL START DATE  START TIME  COUNT HOST SESSION
REQ   INTREQ
-----
ARM  (118)ARM_2          1 2018/08/03  16:10:00.00    0    0    0
0    0
ARM  (118)ARM_2          2 2018/08/03  16:30:00.00    2    1  5909

```

```

4      4
ARM (118)ARM_2      3 2018/08/03 16:40:00.00      8      1      6184
3      3
ARM (118)ARM_2      4 2018/08/03 16:50:00.00      8      1      6676
3      3
ARM (118)ARM_2      5 2018/08/03 17:00:00.00      6      1      7138
3      3
ARM (118)ARM_2      6 2018/08/03 17:10:00.00      2      1      7366
3      3
ARM (117)ARM_1      1 2018/08/03 17:10:30.00     40      1      7334
4      4
ARM (117)ARM_1      2 2018/08/03 17:10:40.00     17      1      7355
3      3
ARM (117)ARM_1      3 2018/08/03 17:10:50.00      0      0          0
0      0
ARM (117)ARM_1      4 2018/08/03 17:11:00.00      0      0          0
0      0
ARM (117)ARM_1      5 2018/08/03 17:11:10.00      0      0          0
0      0
ARM (117)ARM_1      6 2018/08/03 17:11:20.00     33      1      7362
4      4

```

## TDWM SUMMARY

Returns the Teradata dynamic workload management software WD summary data fields.

### Input Data

| Field Name        | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | <p>Indicator bits that specify which fields to treat as NULL if you are using indicator mode.</p> <p>Each bit in the byte corresponds to one field in the input data.</p> <p>If data is supplied for that field, set the bit to zero.</p> <p>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.</p> <p><b>Note:</b></p> <p>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode.</p> |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | <p>MONITOR software version ID. This can be version 6 or later.</p> <p>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a>.</p>                                                                                                                                                                                                                                                                                                                |



## Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

There is one record per WD active in the system. The record contains counts of the number of queries that were classified into the WD in the collection period and the number of queries that completed in the collection period along with query statistics.

The TDWM Summary request returns information for all queries completed in the dashboard interval for the WDs.

If TASM Workloads are not enabled, only the first statement is returned.

## CLiv2 Response Parcels

The TDWM SUMMARY request is treated internally as a two statement request, with each statement generating a response. The two statement response returned from the database contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                     | Comments/Key Parcel Body Fields                                                                                               |
|-----------------|---------------|--------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273                                                                                                          | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = 156 (PCLTWMSUMMARYSTMT)                                                |
| DataInfo        | 71            | 6 to 64100                                                                                                         | Optional: This parcel is present if request was IndicData parcel.                                                             |
| Record          | 10            | <ul style="list-style-type: none"> <li>• 5 to 64100(record mode)</li> <li>• 6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is returned in record or indicator mode. This is the only record returned. |
| EndStatement    | 11            | 6                                                                                                                  | StatementNo = 2-byte integer                                                                                                  |
| Success         | 8             | 18 to 273                                                                                                          | StatementNo = 2<br>ActivityCount = number of record parcels returned<br>ActivityType = 134 (PCLTDWMSUMSTMT)                   |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                  | Comments/Key Parcel Body Fields                                                                                                                                                                         |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DataInfo        | 71            | 6 to 64100                                                                                                      | Optional: This parcel is present if request was IndicData parcel.                                                                                                                                       |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is returned in record or indicator mode. There is one Summary Data record returned for each active WD. The format of this record is described below. |
| EndStatement    | 11            | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                                                                                            |
| EndRequest      | 12            | 4                                                                                                               | None                                                                                                                                                                                                    |

## Response

### Note:

Each of the statement types described below correspond to a ResultSet returned by the Teradata JDBC Driver, and each statement type field corresponds to a ResultSet column. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Statement 1

The response to the first statement results in a Record parcel containing the *SampleSec* field.

| Field/Column Name | Data Type | Description                                                                               |
|-------------------|-----------|-------------------------------------------------------------------------------------------|
| SampleSec         | SMALLINT  | Duration of the collection period, in seconds. This value represents the collection rate. |

## Statement 2

The response to the second statement results in a Record parcel that consists of summary information for an active WD in the system collected in the last collection period. There is one record for each WD.

### Note:

If TASM Workloads are not enabled, this statement is not returned.

The following table describes the format of the Summary Data Record parcel.

| Field/Column Name | Data Type | Description |
|-------------------|-----------|-------------|
| WD ID             | INTEGER   | WD ID.      |

| Field/Column Name     | Data Type | Description                                                                                                                                                                                                                                                                 |
|-----------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Arrivals              | INTEGER   | Number of queries that were classified into this WD by the Teradata dynamic workload management software.                                                                                                                                                                   |
| Completions           | INTEGER   | Number of queries that completed in this WD in this dashboard or logging interval with: <ul style="list-style-type: none"> <li>• No error code or an error code of 3158</li> <li>• No abort flag</li> <li>• Either an AMP count of &gt; 0 or a Parser CPU &gt; 0</li> </ul> |
| Minimum Response Time | FLOAT     | Minimum response time in centiseconds of any query.                                                                                                                                                                                                                         |
| Maximum Response Time | FLOAT     | Maximum response time in centiseconds for any query.                                                                                                                                                                                                                        |
| Average Response Time | FLOAT     | Average response time in centiseconds for this workload based on the total response time of all completions divided by number of completions.                                                                                                                               |
| Minimum CPU Time      | FLOAT     | Minimum CPU in milliseconds of all the queries that completed in this workload in this dashboard or logging interval, where <i>CPU</i> is the minimum CPU used by any one AMP of the query plus the Parser CPU time used by that same query.                                |
| Maximum CPU Time      | FLOAT     | Maximum CPU in milliseconds of all the queries that completed in this workload in this dashboard or logging interval, where <i>CPU</i> is the maximum CPU used by any one AMP of the query plus the Parser CPU time used by that same query.                                |
| Average CPU Time      | FLOAT     | Running total in milliseconds of Parser CPU time plus the total AMP CPU of each query that completed in this workload divided by the number of completions of queries in this workload in this dashboard or logging interval.                                               |
| Delayed Count         | INTEGER   | Number of queries that are delayed in this WD.                                                                                                                                                                                                                              |
| Average Delay Time    | FLOAT     | Average delay time in centiseconds for all queries assigned to this WD.                                                                                                                                                                                                     |
| Exception Count       | INTEGER   | Number of queries with exceptions in this WD.                                                                                                                                                                                                                               |
| Met SLG Count         | INTEGER   | Number of queries that met WD Response Time SLG requirements.<br><br><b>Note:</b><br>If the WD does not have Response Time SLG requirement, the query is still counted as met.                                                                                              |
| Active Query Count    | INTEGER   | Number of active queries assigned to this WD.                                                                                                                                                                                                                               |
| Active Delayed Count  | INTEGER   | Number of currently delayed queries in this WD.                                                                                                                                                                                                                             |
| Error Count           | INTEGER   | Number of queries that finished in this WD in this dashboard or logging interval with an error code > 0 but not 3158 or 3156.                                                                                                                                               |

| Field/Column Name | Data Type | Description                                                                                                                                                                                                               |
|-------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Abort Count       | INTEGER   | Number of queries that finished in this WD in this dashboard or logging interval with an error code of 0 or 3156 and an abort flag.                                                                                       |
| Other Count       | INTEGER   | Number of queries that finished in this WD in this dashboard or logging interval with no: <ul style="list-style-type: none"> <li>• Error code</li> <li>• Abort flag</li> <li>• AMP count</li> <li>• Parser CPU</li> </ul> |
| CollectionDate    | DATE      | Date of the collection of summary information as pulled from the StartCollectTime field on the control vproc.                                                                                                             |
| CollectionTime    | FLOAT     | Time of the collection of summary information as pulled from the StartCollectTime field on the control vproc.                                                                                                             |
| ExceptionAbCnt    | INTEGER   | Number of queries that aborted due to a TDWM exception.                                                                                                                                                                   |
| ExceptionMvCnt    | INTEGER   | Number of queries that were moved into this WD due to an exception.                                                                                                                                                       |
| ExceptionCoCnt    | INTEGER   | Number of queries that hit an exception but continued in this WD.                                                                                                                                                         |
| IntervalDelayCnt  | INTEGER   | Number of queries that were delayed in this WD during this interval.                                                                                                                                                      |
| RejectedCount     | INTEGER   | Number of queries that were rejected by TDWM due to a filter or throttle rule violation.                                                                                                                                  |
| MovedInCount      | INTEGER   | Number of queries that were moved into this WD either due to an exception action or manual move via PMPC.                                                                                                                 |
| VirtualPartNum    | INTEGER   | Virtual Partition number of the WD.                                                                                                                                                                                       |
| AvgIOWaitTime     | FLOAT     | Average duration of sleeps in milliseconds due to the I/O rate handling over the life of all requests completing in this WD during this interval.                                                                         |
| MaxIOWaitTime     | FLOAT     | Maximum duration of sleeps in milliseconds due to the I/O rate handling of a request completing in this WD during this interval.                                                                                          |
| AvgOtherWaitTime  | FLOAT     | <b>Note:</b><br>This field is reserved for future use.                                                                                                                                                                    |
| MaxOtherWaitTime  | FLOAT     | <b>Note:</b><br>This field is reserved for future use.                                                                                                                                                                    |
| AvgCPURunDelay    | FLOAT     | Average wait time in milliseconds in the run queue of all the requests completing in this WD during this interval.                                                                                                        |
| MaxCPURunDelay    | FLOAT     | Maximum wait time in milliseconds in the run queue of a request completing in this WD during this interval.                                                                                                               |

| Field/Column Name  | Data Type | Description                                                                                                                                                                             |
|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AvgSeqRespTime     | FLOAT     | Average of the sum of the response times of the steps (as if all steps had been executed sequentially) of all the requests completing in this WD during this interval, in milliseconds. |
| MaxSeqRespTime     | FLOAT     | Maximum sum of the response times of the steps (as if all steps had been executed sequentially) of a request completing in this WD during this interval, in milliseconds.               |
| AvgLogicalIO       | FLOAT     | Average count of logical I/Os of all requests completing in this WD during this interval.                                                                                               |
| MaxLogicalIO       | FLOAT     | Maximum count of logical I/Os for a request completing in this WD during this interval.                                                                                                 |
| AvgLogicalKBs      | FLOAT     | Average logical I/O in KB of all requests completing in this WD during this interval.                                                                                                   |
| MaxLogicalKBs      | FLOAT     | Maximum logical I/O usage in KB of all requests completing in this WD during this interval.                                                                                             |
| AvgPhysicalIO      | FLOAT     | Average count of physical I/Os of all requests completing in this WD during this interval.                                                                                              |
| MaxPhysicalIO      | FLOAT     | Maximum physical I/O usage of all requests completing in this WD during this interval.                                                                                                  |
| AvgPhysicalKBs     | FLOAT     | Average physical I/O usage in KB of all requests completing in this WD during this interval.                                                                                            |
| MaxPhysicalKBs     | FLOAT     | Maximum physical I/O usage in KB for a request completing in this WD during this interval.                                                                                              |
| ThrottleBypassed   | INTEGER   | The number of queries reported in the Completions field that were allowed to run due to the TASM ThrottleBypass ruleset attribute.                                                      |
| FlexActive         | INTEGER   | The number of queries that are currently active in the system that were released by the Flex Throttle feature.                                                                          |
| FlexComplete       | INTEGER   | The number of queries that were released by the Flex Throttle feature that completed during this period.                                                                                |
| FlexArrivals       | INTEGER   | The number of new requests that became active due to Flex Throttle feature.                                                                                                             |
| Defer Count        | INTEGER   | Number of queries that are deferred in this WD due to an arrival rate meter rule.                                                                                                       |
| Avg Defer Time     | FLOAT     | Average defer time due to arrival rate meter rules in centiseconds for all queries assigned to this WD.                                                                                 |
| Active Defer Count | INTEGER   | Number of currently deferred queries in this WD.                                                                                                                                        |

## Teradata Vantage™ - Application Programming Reference, Release 17.10

350

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size.

```

SUCCESS parcel:
StatementNo=1,      ActivityCount=1,
ActivityType=156, FieldCount=1
TDWM Summary Request successful.
      Sample Seconds: 120
SUCCESS parcel:
StatementNo=2,      ActivityCount=5,
ActivityType=156, FieldCount=43
TDWM Summary - # of WDIDs: 5
  WLC  Arrivals  Complete  WereDlyd  Exceptns  Met  SLG  CurrActv  CurrDlyd  Abt
Cnt   Othr Cnt
      ID   Min Resp  Max Resp  Avg Resp  Min CPU  Max CPU  Avg CPU  AvgDelay  Err Cnt
      ID   Min Resp  Max Resp  Avg Resp  Min CPU  Max CPU  Avg CPU  AvgDelay  Err Cnt
      ID   Min Resp  Max Resp  Avg Resp  Min CPU  Max CPU  Avg CPU  AvgDelay  Err Cnt
-----

```

```

12      0      0      0      0      0      1      0
0      0
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0      2011/06/15 18:33:49.00
13      0      0      0      0      0      0      0
0      0
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0      2011/06/15 18:33:49.00
14      0      0      0      0      0      0      0
0      0
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0      2011/06/15 18:33:49.00
15      0      0      0      0      0      0      0
0      0
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0      2011/06/15 18:33:49.00
16      0      0      0      0      0      0      0
0      0
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0      2011/06/15 18:33:49.00

```

## TDWM WD ASSIGNMENT

Changes the Workload a session or request is assigned to.

### Input Data

| Element           | Data Type           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode. |
| <i>mon_ver_id</i> | SMALLIN<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.<br>For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                               |
| <i>host_id</i>    | SMALLINT            | Host upon which the session was established. The host ID cannot exceed 1023. A host ID of zero identifies the database operator console.                                                                                                                                                                                                                                                                                                                  |
| <i>session_no</i> | INTEGER             | Session number for his record. This value is assigned by the host (or client) at logon.<br>Together with a given Host ID, a session number uniquely identifies a session on the database system.                                                                                                                                                                                                                                                          |

| Element            | Data Type | Description                                                                                                                 |
|--------------------|-----------|-----------------------------------------------------------------------------------------------------------------------------|
| <i>PE_vproc_no</i> | SMALLINT  | PE vproc number where the session runs.                                                                                     |
| <i>scope</i>       | SMALLINT  | Scope of the change defined: <ul style="list-style-type: none"> <li>• 0 = Current request</li> <li>• 1 = Session</li> </ul> |
| <i>WLC_id</i>      | INTEGER   | WD ID to which the request should be assigned.                                                                              |

## Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

The TDWM WD ASSIGNMENT request is comparable to changing the session priority by the way of the SQL SET SESSION ACCOUNT statement when TASM Workloads are not enabled.

Use TDWM WD ASSIGNMENT to change the assigned WD for an individual request or session in cases where the rules imposed by the current WD prevent it from running as needed.

Use the MONITOR SESSION request to determine *host\_id*, *session\_no*, *PE\_vproc\_no*, *scope*, and *WLC\_id* for the TDWM WD ASSIGNMENT input.

### Note:

You must enable TASM Workloads in the Teradata Viewpoint Workload Designer portlet to use this request.

The following table shows how the *scope* determines the effects of the TDWM WD ASSIGNMENT.

| If the scope = ...  | And there is ...                                                                                                                                                          |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 (current request) | no active request, this request has no effect.                                                                                                                            |
| 0 (current request) | an active request, the current active request is moved into the specified WD. The “change workload” exception if specified is ignored for this request.                   |
| 1 (session) request | no active request, all subsequent requests on this session are assigned into the specified WD. If specified, the “change workload” exception is ignored for all requests. |



| If the scope = ... | And there is ...                                                                                                                                                                                     |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 (session)        | an active request, the current active request, and all subsequent requests on the session, move into the specified WD.<br>If specified, the “change workload” exception is ignored for all requests. |

### CLiv2 Request Parcels

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                                                     |
|-----------------|---------------|----------------|-------------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273      | StatementNo = 1<br>ActivityCount = 0<br>ActivityType = 159 (PCLTWMWDASSIGNMENTSTMT) |
| EndStatement    | 11            | 6              | StatementNo = 1                                                                     |
| EndRequest      | 12            | 4              | None                                                                                |

### Sample Input - CLiv2 Request

The following example shows how the parcels for a TDWM WD ASSIGNMENT request, built by CLiv2, appear when sent to the database server.

#### Note:

In this example, the size of the response buffer in the example is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body                                                                |                                   |
|--------|------|--------|---------------------------------------------------------------------|-----------------------------------|
| Num    | Name | Bytes  | Field                                                               | Value                             |
| 0001   | Req  | 16     | Request                                                             | TDWM WD ASSIGNMENT                |
| 0003   | Data | 8      | MonVerID<br>host_id<br>session_no<br>PE_vproc_no<br>scope<br>WLC_id | 7<br>1<br>1000<br>16383<br>1<br>3 |
| 0004   | Resp | 6      | BufferSize                                                          | 64000                             |

## Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Sample Output

The TDWM WD ASSIGNMENT request returns values approximately as shown below when TASM Workloads rule is enabled and the following input data is specified:

- *host\_id* = 1
- *session\_no* = 1000
- *PE\_vproc\_no* = 16383
- *scope* = 1
- *WLC\_id* = 3

The TDWM WD ASSIGNMENT request commonly returns values in text character format. Your application program may return the values in a different format or display.

```
Success parcel:
StatementNo: 1    ActivityCount: 1
ActivityType: 159    FieldCount: 1
DataInfo parcel:
FieldCount: 1
EndStatement.
EndRequest.
```

## USER EVENT CONTROL

Activates and deactivates a user event.

### Input Data

| Element           | Data Type | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE      | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLlv2 request is submitted in indicator mode. |
| <i>mon_ver_id</i> | SMALLINT  | MONITOR software version ID. This can be version 6 or later.                                                                                                                                                                                                                                                                                                                                                                                                  |

| Element             | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | NOT NULL             | For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                                                                                                                                       |
| <i>Request Type</i> | SMALLINT<br>NOT NULL | Type of request specified: <ul style="list-style-type: none"> <li>• 0 = Deactivate event identified in <i>Event Name</i></li> <li>• 1 = Activate event identified in <i>Event Name</i></li> </ul>                                                                                                                                                                                                                                                 |
| <i>Event Name</i>   | VARCHAR<br>(30)      | The name of the user event. The event name is a maximum of 30 characters and must be padded with blanks. It is also case sensitive.                                                                                                                                                                                                                                                                                                               |
| <i>Duration</i>     | INTEGER<br>NOT NULL  | Time, in minutes, that this user event remains active. This field is valid for active requests only.<br>A value of zero means the event is true.<br><br><b>Note:</b><br>The evaluation of all events, including user events, is based on the System Event Timer defined in the Teradata Dynamic Workload Designer portlet. If the Duration value is not a multiple of the System Event timer, it is rounded to the next System Event timer value. |

## Monitor Privileges

To use this request, you must have the ABORTSESSION and MONSESSION privileges as part of your default role or both privileges must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

User Events are defined externally only by the name and the attribute of being either active or inactive. User Events also have an optional duration time that determines the length of time the Activate event persists. The event times out at the end of the duration time unless it is de-activated or re-activated. A *Duration* value of zero indicates the Activate (a *Request Type* value of 1) is persistent and lasts until explicitly made inactive by another User Event call. User Events persist across a TPA restart or system failure.

User Event names are global, or system wide and, therefore, you must make sure that no conflicting usages of User Event Control request calls occur.

## CLIV2 Response Parcels

The following table lists information about the parcels for the USER EVENT CONTROL request.

| Parcel Sequence | Flavor | Field Length                                                                                                    | Comments and Key Parcel Body Fields                                                                                          |
|-----------------|--------|-----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| Success         | 8      | 18 to 273                                                                                                       | StatementNo=1<br>ActivityCount = 1<br>ActivityType =<br>PCLUSEREVENTCONTROLSTMT (169)                                        |
| DataInfo        | 71     | 6 to 64100                                                                                                      | Optional: this parcel is present if request was IndicData parcel.                                                            |
| Record          | 10     | <ul style="list-style-type: none"> <li>5 to 64100 (record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData) data is returned in record or indicator mode. This is the only record returned. |
| EndStatement    | 11     | 6                                                                                                               | StatementNo = 2-byte integer                                                                                                 |
| EndRequest      | 12     | 4                                                                                                               | None                                                                                                                         |

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The response to the USER EVENT CONTROL request results in a single Record parcel containing the result of the request.

| Field/Column Name                  | Data Type        | Description                                                                                                                                                                             |
|------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User-Defined Event Status          | INTEGER NOT NULL | Current status of the user-defined event: <ul style="list-style-type: none"> <li>0 = Inactive</li> <li>1 = Active</li> </ul>                                                            |
| Previous User-Defined Event Status | INTEGER NOT NULL | Previous status of the user-defined event: <ul style="list-style-type: none"> <li>0 = Previously inactive</li> <li>1 = Previously active</li> <li>2 = Previously not defined</li> </ul> |

The USER EVENT CONTROL request can be used to activate or inactivate a single, specific User Event.

The response indicates if the operation was successful. For successful operations, the result (Previous User-Defined Event Status) shows the previous value of this User Event. This allows the user to know if the User Event was previously active or not. For example, a successful operation, can return any of the following results:

| The result.... | indicates...                                                        |
|----------------|---------------------------------------------------------------------|
| 1,0            | the current User Event is Active and it was previously Inactive.    |
| 1,0            | the current User Event is Inactive and it was previously Active.    |
| 1,2            | the current User Event is Active and it was previously not defined. |

**Note:**

This table only describes some of the possible response combinations, there are others.

## Open APIs (SQL Interfaces)

This section describes the Teradata Dynamic Workload Management SQL interfaces that are used to manage the workload and update the stored components (such as, WDs and related data) in the Teradata Dynamic Workload Management. These SQL interfaces consist of UDFs and external stored procedures that you can invoke from any application.

The SQL interfaces to workload management are installed by the DIPTDWM script as part of the database installation (see [Requirements for Using the API](#)).

These SQL interfaces provide similar functionality to the Teradata Dynamic Workload Management CLIV2 or Teradata JDBC Driver requests.

**Note:**

When issuing Teradata dynamic workload management software UDFs or calling Teradata dynamic workload management software external stored procedures, you must fully qualify them by the database name, TDWM.

For examples of Teradata dynamic workload management software functions and external stored procedures, see the following topics.

## TDWMAbortDelayedRequest

Aborts a request or utility session on the Teradata dynamic workload management software defer or delay queue.

This function returns a zero if it is successful.

## Syntax

```
REPLACE FUNCTION TDWM.TDWMAbortDelayedRequest (
  HostId SMALLINT,
  SessionNo INTEGER,
  RequestNo INTEGER,
  WDId INTEGER
) RETURNS INTEGER
...
;
```

## Syntax Elements

### *HostId*

Host ID for the session.

### *SessionNo*

Number of the session.

### *RequestNo*

Request number of the task.

If the value is zero, the utility session is aborted.

### *WDId*

WD ID for the request in the defer or delay queue being examined.

## Usage Notes

The TDWMAbortDelayedRequest function provides similar functionality to a TDWM DELAY REQUEST CHANGE request. For more information, see [TDWM DELAY REQUEST CHANGE](#).

### Example: Using TDWMAbortDelayedRequest

This example shows how to abort a delayed request on *SessionNo* 1011.

```
SELECT TDWM.TDWMAbortDelayedRequest(HostId, SessionNo, RequestNo, 0)
FROM TABLE (TDWMGetDelayedQueries('O')) AS t1
WHERE SessionNo=1011;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 3 seconds.
TDWMAbortDelayedRequest(HostId,SessionNo,RequestNo,0)
```

-----  
0

Assuming the request on *SessionNo* 1011 was delayed, it is aborted as follows:

```
SELECT DatabaseName FROM dbc.databasesv;
*** Starting at Tue Apr 26 15:22:42 2005
*** Failure 3151 TWM Workload violation: Delay Request Change Abort
        Statement# 1, Info =0
```

## TDWMActiveWDs

Returns a list of all active workloads. TDWMActiveWDs is a table function.

### Syntax

```
REPLACE FUNCTION TDWMActiveWDs (
) RETURNS TABLE (
    WDIId INTEGER,
    WDName VARCHAR(30) CHARACTER SET UNICODE,
    Status VARCHAR(8) CHARACTER SET LATIN,
    SLG INTEGER,
    PGID INTEGER,
    AGID INTEGER,
    LogMode VARCHAR(8) CHARACTER SET LATIN,
    CurrOpEnv INTEGER,
    ActualOpEnv INTEGER
)
LANGUAGE C
NO SQL
NO EXTERNAL DATA
PARAMETER STYLE SQL
NOT DETERMINISTIC
CALLED ON NULL INPUT
EXTERNAL NAME 'SL!api';
```

### Syntax Elements

#### WDId

The WD number.

#### WDName

The WD name.

**Status**

Enabled status.

**SLG**

Service level goal.

**PGID**

Performance group number assigned to the WD.

**AGID**

Allocation group number assigned to the WD.

**LogMode**

Type of logging.

**CurrOpEnv**

Current operating environment.

**ActualOpEnv**

Active operating environment.

**Usage Notes**

The GDO for TDWM.TDWMActiveWDS exceeds the size of the protected buffer, so this function uses the non-protected buffer. Note, this means the function is not protected.

**Example: Using TDWMActiveWDS**

```
sel * from table (tdwm.TDWMActiveWDS()) as t1;
```

```
*** Query completed. One row found. 9 columns returned.
```

```
*** Total elapsed time was 1 second.
```

| WDId | WDName     | Status  | SLG  | PGID | AGID  | LogMode | CurrOpEnv | ActualOpEnv |
|------|------------|---------|------|------|-------|---------|-----------|-------------|
| ---- | -----      | -----   | ---- | ---- | ----- | -----   | -----     | -----       |
| 250  | WD-Default | Default | 0    | 0    | 0     | Summary | 12        | 12          |

**TDWMApply**

Applies changes to one or more of the Teradata dynamic workload management software categories.



## Syntax

```
REPLACE PROCEDURE TDWM.TDWMApply (
  IN RuleSetID INTEGER,
  IN FilterCat VARCHAR(1) CHARACTER SET LATIN,
  IN ThrottleCat VARCHAR(1) CHARACTER SET LATIN,
  IN WorkloadCat VARCHAR(1) CHARACTER SET LATIN,
  IN EventCat VARCHAR(1) CHARACTER SET LATIN
)
  ...
;
```

## Syntax Elements

### *RuleSetID*

ID of the rule set to apply. This must be the current rule set.

### *FilterCat*

Request for the Filters rule category:

- Y = Apply changes to the rule category
- N = Skip this rule category

### *ThrottleCat*

Request for the system query or system session limits rule category:

- Y = Apply changes to the rule category
- N = Skip this rule category

### *WorkloadCat*

Request for the workloads rule category:

- Y = Apply changes to the rule category
- N = Skip this rule category

### *EventCat*

Request for the Event category:

- Y = Apply changes to the category
- N = Skip this category

## Usage Notes

Do not issue TDWMAppl while holding locks on any TDWM database tables. This causes a self-deadlock condition because the database must read the TDWM database tables to perform this apply.

Using the TDWMAppl procedure, you can only apply changes to the active rule set.

This procedure activates all rules in the database for the category being applied.

## Example: Using TDWMAppl

```
CALL TDWM.TDWMAppl (200, 'Y', 'N', 'N', 'N');
```

## TDWMAssignWD

Changes the workload a session or request it is assigned to.

This function returns a zero if it is successful.

## Syntax

```
REPLACE FUNCTION TDWM.TDWMAssignWD (
  HostId SMALLINT,
  SessionNo INTEGER,
  VprocNo INTEGER,
  Scope VARCHAR(1),
  WDIId INTEGER
) RETURNS INTEGER
...
;
```

## Syntax Elements

### HostId

ID of the host upon which the session was issued. *HostId* cannot exceed 1023. A *hostid* of zero identifies the database operator console.

### SessionNo

Session number. *SessionNo* combined with *HostId* produces a unique session ID.

### VprocNo

PE Vproc number where the session runs.

### Scope

Scope of the change defined:

- R = Change the workload of the current request
- S = Change the workload of the session

**WDId**

WD ID to which the request should be assigned.

**Usage Notes**

If TASM Workloads are not enabled and if the database cannot find a valid host ID and session number combination, an error message is returned.

The TDWMAssignWD function provides similar functionality to a TDWM WD ASSIGNMENT request. For more information about this interface, see [TDWM WD ASSIGNMENT](#).

**Example: Using TDWMAssignWD**

```
SELECT TDWM.TDWMAssignWD(Hostid, sessionNo, runvprocno, 'S', 7)
FROM TABLE (MonitorSession(1,'*',0)) AS t1
where username='user14';
*** Query completed. 5 rows found. One column returned.
*** Total elapsed time was 2 seconds.
TDWMAssignWD(HostId,SessionNo,RunVprocNo,'S',7)
-----
0
0
0
0
0
```

**TDWMEventControl**

Activates or deactivates a user-defined event.

**Syntax**

```
REPLACE PROCEDURE TDWM.TDWMEventControl (
  IN EventName TD_ANYTYPE,
  IN Operation VARCHAR(1) CHARACTER SET LATIN,
  IN Duration INTEGER,
  OUT NewStatus VARCHAR(8) CHARACTER SET LATIN,
  OUT PriorStatus VARCHAR(8) CHARACTER SET LATIN
)
...
;
```

## Syntax Elements

### *EventName*

Name of the event specified.

### *Operation*

Status of the event:

- A = Active
- I = Inactive

### *Duration*

Time, in minutes, after the event expires.

A value of zero indicates no expiration time.

### *NewStatus*

Status of the event after the procedure call.

### *PriorStatus*

Status of the event before the procedure call.

## Usage Notes

The user-defined event specified must be an existing event in the active rule set and can include the following actions:

- Notifications (for example, send alerts, run programs, or post to a queue table)
- Activate a health condition (SysCon) or planned environment (OpEnv). The highest priority for health conditions and planned environments determines the rule state values that the system enforces. Rule attributes that can be changed based on the rule state include:
  - Rule enable flag
  - Rule system session, system query, and utility throttle limits
  - Workload throttle limits

For more information on SysCon and OpEnv, see the Teradata Viewpoint Help.

The TDWMEventControl procedure provides similar functionality to the USER EVENT CONTROL request. For more information about this interface, see [USER EVENT CONTROL](#).

### Example: Using TDWMEventControl

```
CALL TDWM.TDWMEventControl('NodeDownEvent', 'A', 0, NewStatus, PriorStatus);
```

## TDWMEventMapping

Lists all objects that make up the system state.

### Syntax

```
REPLACE FUNCTION TDWM.TDWMEventMapping
  RETURNS TABLE (
    ObjectName VARCHAR(128) CHARACTER SET UNICODE,
    ObjectType VARCHAR(10) CHARACTER SET LATIN,
    ObjectId INTEGER,
    ObjectActive CHAR CHARACTER SET LATIN,
    PrecSeverity INTEGER,
    EventKind VARCHAR(12) CHARACTER SET LATIN,
    EventClass VARCHAR(12) CHARACTER SET LATIN,
    StatusDate INTEGER,
    StatusTime FLOAT
  )
  ...
;
```

### Syntax Elements

#### ObjectType

Type of object, for example: an Event, Expression, SysCon, OpEnv, and a State.

#### ObjectId

Internal object ID number.

#### ObjectName

Type of object, for example: an Event, Expression, SysCon, OpEnv, and a State.

#### ObjectActive

Status of the object:

- 0 = Inactive
- 1 = Active

#### PrecSeverity

Indicator of the Precedence or Severity of the SysCon or OpEnv object types only:

- SysCon = Indicates the Severity of the SysCon.

- OpEnv = Indicates the Precedence of the OpEnv.

## EventKind

Text string of the specific event. You may see the following strings as a result of this interface:

- User Defined SysCon. The user-defined system condition.
- User Defined OpEnv. The user-defined operating environment.
- Time Period. The time range.
- Fatal AMPs. The number of AMPs with the status of FATAL.
- Fatal PEs. The number of PEs with the status of FATAL.
- Fatal GTWs. The number of Gateways with the status of FATAL.
- Nodes Down. The % of nodes down within a clique.
- Minimum available AWTs. The number of in-use AMP AWTs.
- AMPs In Flow Control. The number of AMPs in flow control.
- Average System CPU. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables and compared to the threshold defined by the user in this event. For more information, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.
- System CPU Skew. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables. For more information, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.
- WD CPU %. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables. For more information, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.
- WD SLG Response. The % of request in this WD that have met the defined Response Time Service Level Goal (SLG).
- WD SLG Throughput. The % of request in this WD that have met the defined Throughput SLG.
- WD Arrivals. The number of new requests for this WD.
- WD Active Request. The number of currently active requests in this WD.
- WD Delay Queue Depth. The number of requests on the Delay Queue for this WD.
- WD Delay Query Time. The max time a request has been delayed in this WD.
- WD AWT Wait Time. The max time a request was on an AMP mailbox waiting for an AWT.
- TwmFlexAvailableAWTsEvent. The number of AWTs that are available for work.
- TwmFlexAvgCpuEvent. This value is calculated by the CPU usage columns in the ResUsageSpma and ResUsageSps tables and compared to the threshold defined by the user in this event. For more information, see *Teradata Vantage™ - Resource Usage Macros and Tables*, B035-1099.

**EventClass**

This field is only valid for Object Type of Event and Expression. The values are:

- 1 = OpEnv
- 2 = SysCon
- 3 = FlexThrtl

**Usage Notes**

The TDWMEventMapping function provides similar functionality to an EVENT STATUS request. For more information about this interface, see [EVENT STATUS](#).

**Example: Using TDWMEventMapping**

```
SELECT ObjectName, ObjectType, ObjectId, ObjectActive
FROM TABLE (TDWM.TDWMEventMapping()) AS t1
where objecttype='syscon';
*** Query completed. 3 rows found. 4 columns returned.
*** Total elapsed time was 1 second.
```

| ObjectName      | ObjectType | ObjectId | ObjectActive |
|-----------------|------------|----------|--------------|
| -----           | -----      | -----    | -----        |
| Extra SysCon1   | SYSCON     | 225      | I            |
| Degraded SysCon | SYSCON     | 250      | I            |
| Normal          | SYSCON     | 200      | A            |

**TDWMEventStatus**

Returns the currently defined events.

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMEventStatus (
  RequestType VARCHAR(1) CHARACTER SET LATIN
) RETURNS TABLE (
  EventId INTEGER,
  EventStatus CHAR(1) CHARACTER SET LATIN,
  EventActiveDate INTEGER,
  EventActiveTime FLOAT,
  ExpressionId INTEGER,
  ExpressionStatus CHAR(1) CHARACTER SET LATIN,
  ExpActiveDate INTEGER,
  ExpActiveTime FLOAT,
  RecordType VARCHAR(8) CHARACTER SET LATIN,
  RecordId INTEGER,
```

```

RecordStatus CHAR(1) CHARACTER SET LATIN,
DueToDuration CHAR(1) CHARACTER SET LATIN,
ActiveDate INTEGER,
ActiveTime FLOAT
)
...
;

```

## Syntax Elements

### *RequestType*

Request type:

- A= Return all events.
- C = Return current state events.

### **EventId**

Internal ID of the Event. A value of zero indicates there is no event for this record entry and there are no events for the Normal SysCon, the Always OpEnv and a State.

Events are reported in the order they appear within the associated expression. Only events in the associated expression are included.

### **EventStatus**

Current status of the event. The value is either Inactive or Active.

### **EventActiveDate**

Date when EventStatus was last changed. This is the date when the EventStatus was set to Inactive or Active.

### **EventActiveTime**

Time when EventStatus was last changed. This is the time when EventStatus is set to Inactive or Active.

### **ExpressionId**

Internal ID of the Expression. A value of zero indicates there is no expression for this record entry.

There are no expressions for the Normal SysCon, the Always OpEnv, and a State.

For the CURRENT state request, the first TRUE expression for the State is included. When there are multiple TRUE expressions for a state, only the first one processed is reported. The



order of processing is consistent within a rule set and, therefore, the same TRUE expression is reported on every request.

**ExpressionStatus**

Current status of the Expression:

- 0 = Inactive
- 1 = Active

**ExpActiveDate**

Date when ExpressionStatus last changed.

**ExpActiveTime**

Time when Expression Status was last changed. This is the time when Expression Status was set to either Active or Inactive.

**RecordType**

Type of record, for example: a SysCon, OpEnv, or Notify Only Expression.

**RecordId**

Internal ID of the SysCon or OpEnv depending on the type of record specified in the RecordType field.

A value of zero indicates the Notify Only Expression record type.

**RecordStatus**

Current status of the SysCon or OpEnv record:

- Y = Active
- N = Inactive

**DueToDuration**

An indicator that the corresponding Syscon is being considered as TRUE or is TRUE:

- 0 = Inactive if the SysCon is TRUE based on the defined expressions.
- 1 = Active if the corresponding SysCon is being considered as TRUE due to the MinDuration value being in affect.

This field is valid only when SysCon is the specified RecordType.

**StateActiveDate**

Date when RecordStatus was last changed.

**StateActiveTime**

Time when RecordStatus was last changed.

**Usage Notes**

The TDWMEventStatus function provides similar functionality to an EVENT STATUS request. For more information about this interface, see [EVENT STATUS](#).

**Example: Using TDWMEventStatus**

```
SELECT * FROM TABLE (TDWM.TDWMEventStatus('c')) AS t1;
*** Query completed. 3 rows found. 14 columns returned.
*** Total elapsed time was 1 second.
```

|                  |                        |
|------------------|------------------------|
| EventId          | 0                      |
| EventStatus      | I                      |
| EventActiveDate  | 0                      |
| EventActiveTime  | 0.000000000000000E 000 |
| ExpressionId     | 0                      |
| ExpressionStatus | I                      |
| ExpActiveDate    | 0                      |
| ExpActiveTime    | 0.000000000000000E 000 |
| RecordType       | SYSCON                 |
| RecordId         | 1                      |
| RecordStatus     | A                      |
| DueToDuration    | N                      |
| ActiveDate       | 1070518                |
| ActiveTime       | 8.393800000000000E 004 |
| EventId          | 0                      |
| EventStatus      | I                      |
| EventActiveDate  | 0                      |
| EventActiveTime  | 0.000000000000000E 000 |
| ExpressionId     | 0                      |
| ExpressionStatus | I                      |
| ExpActiveDate    | 0                      |
| ExpActiveTime    | 0.000000000000000E 000 |
| RecordType       | OPENV                  |
| RecordId         | 1                      |
| RecordStatus     | A                      |
| DueToDuration    | N                      |
| ActiveDate       | 1070518                |
| ActiveTime       | 8.393800000000000E 004 |
| EventId          | 0                      |
| EventStatus      | I                      |
| EventActiveDate  | 0                      |

```

EventActiveTime    0.000000000000000E 000
  ExpressionId      0
ExpressionStatus    I
  ExpActiveDate      0
  ExpActiveTime      0.000000000000000E 000
  RecordType         STATE
    RecordId         1
  RecordStatus       A
  DueToDuration       N
    ActiveDate        1070517
    ActiveTime        1.000060000000000E 005

```

## TDWMExceptionRate

Returns the first record of the TDWM Exception request, which is the collection rate.

The SMALLINT data type returns the duration of the collection period in seconds. This value represents the collection rate.

### Syntax

```

REPLACE FUNCTION TDWM.TDWMExceptionRate (
) RETURNS SMALLINT
...
;

```

### Example: Using TDWMExceptionRate

```

SELECT TDWM.TDWMExceptionRate();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
TDWMExceptionRate()
-----
                        60

```

## TDWMExceptions

Collects the Teradata dynamic workload management software exception data from the database.

### Syntax

```

REPLACE FUNCTION TDWM.TDWMEXCEPTIONS (
) RETURNS TABLE (
  VProcID SMALLINT,
  QueryID BIGINT,

```

```

UserName VARCHAR(128) CHARACTER SET UNICODE,
SessionID INTEGER,
RequestNum INTEGER,
LogicalHostID SMALLINT,
AcctString VARCHAR(128) CHARACTER SET UNICODE,
WDID INTEGER,
OpEnvID INTEGER,
SysConID INTEGER,
ClassifyTime FLOAT,
ClassifyDate INTEGER,
ExceptionTime FLOAT,
ExceptionDate INTEGER,
ExceptionValue INTEGER,
ExceptionAction VARCHAR(10) CHARACTER SET LATIN,
NewWDID INTEGER,
ExceptionCode INTEGER,
ExceptionSubCode INTEGER,
ErrorText VARCHAR(255) CHARACTER SET UNICODE,
ExtraInfo VARCHAR(200) CHARACTER SET UNICODE,
RuleID INTEGER,
WarningOnly VARCHAR(1) CHARACTER SET LATIN,
RejectionCat SMALLINT
)
...
;

```

## Syntax Elements

### VprocID

Vproc the request with the exception was running on.

### QueryID

Query ID for the query that encountered an exception.

### UserName

User name used in the query with the exception.

### SessionID

Session ID of the query with the exception.

### RequestNum

Request number of the query with the exception.

**LogicalHostID**

Logical host ID of the query with the exception.

**AcctString**

Account string of the query with the exception.

**WDID**

WD ID the query with the exception was running in.

**OpEnvID**

Planned environment ID in force when the query encountered the exception.

**SysConID**

Health condition ID in force when the query encountered the exception.

**ClassifyTime**

Time the query with the exception was classified.

**ClassifyDate**

Date the query with the exception was classified.

**ExceptionTime**

Time the query encountered the exception.

**ExceptionDate**

Date the query encountered the exception.

**ExceptionValue**

Type of exception that occurred:

- 0x00000001 - Exception time limit exceeded.
- 0x00000002 - CPU time (AMP and PE) limit exceeded.
- 0x00000004 - Blocked time limit exceeded.
- 0x00000008 - Disk to CPU ratio exceeded.
- 0x00000010 - AMP CPU skew limit exceeded.
- 0x00000020 - AMP I/O count limit exceeded.
- 0x00000040 - AMP I/O skew limit exceeded.
- 0x00000080 - Max row count (for a step) exceeded.
- 0x00000100 - Max row count (for a query) exceeded.
- 0x00000200 - Spool space limit exceeded.

- 0x00000400 - Number of AMPs used in query exceeded.
- 0x00000800 - Disk CPU ratio value exceeded
- 0x00001000 - I/O space value exceeded.

A conversion to hex is used to extract the bit values. For example, a value of 1024 converted to hex is 400.

### ExceptionAction

Exception action taken by the exception handler:

- A =Abort
- C =Change WD  
New WDIId contains the new WD.
- L =Log.
- E =Execute Program  
ExProgram contains the program name.
- T =Alert  
ExAlert contains the alert name.
- N = No action  
This option cannot be combined with other actions. It disables exception detection.
- S =Abort if the statement is a SELECT and no update has been done in the current transaction
- Q = Insert a row to DBC.SystemQTbl

For example, CE stands for change WD and execute program.

### NewWDID

WD the query was moved into if the ExceptionAction was change WD.

### ExceptionCode

Database exception code.

### ExceptionSubCode

Code for additional information.

This field is not currently used.

### ErrorText

Error text generated for the exception.

**ExtraInfo**

Exception values noted at the time of the exception.

**RuleID**

Teradata dynamic workload management software rule ID of the rule with the exception handling directive.

**WarningOnly**

Indicator that this is a warning only.

**RejectionCat**

Teradata dynamic workload management software category this query was rejected from.

**Usage Notes**

The TDWMExceptions function provides similar functionality to a TDWM Exceptions request. For more information about this interface, see [TDWM EXCEPTIONS](#).

**Example: Using TDWMExceptions**

```
SELECT * from table (TDWM.TDWMExceptions()) as t2;
*** Query completed. 1 row found. 24 columns returned.
*** Total elapsed time was 1 second.
      VprocId      16383
      QueryId      163837185173601767
      UserName      LUMBER
      SessionId      32
      RequestNum      2
      LogicalHostId      1
      AcctString      LUMBER
      WdId      11
      OpEnv      1
      SysCon      1
      ClassifyTime      1.55443000000000E 005
      ClassifyDate      1090928
      ExceptionTime      1.55449000000000E 005
      ExceptionDate      1090928
      ExceptionValue      2
      ExceptionAction      LA
      NewWDID      11
      ExceptionCode      3156
      ExceptionSubCode      0
      ErrorText      CPUTime: 1734ms.
```

|              |   |
|--------------|---|
| ExtraInfo    |   |
| RuleId       | 2 |
| WarningOnly  |   |
| RejectionCat | 0 |

## TDWMGetDelayedQueries

Returns the delayed query data fields and delay information.

This table function is only supported in Constant Mode.

### Syntax

```

REPLACE FUNCTION TDWM.TDWMGetDelayedQueries (
  RequestType VARCHAR(1) CHARACTER SET LATIN
) RETURNS TABLE (
  Username          VARCHAR(128) CHARACTER SET UNICODE,
  HostId            SMALLINT,
  SessionNo         INTEGER,
  RequestNo         INTEGER,
  RuleName          VARCHAR(128) CHARACTER SET UNICODE,
  RuleId            INTEGER,
  TotalTimeHeld     INTEGER,
  OverRidable       CHAR CHARACTER SET LATIN,
  BlockingCnt       INTEGER,
  PEId              INTEGER,
  WDDelayed         INTEGER,
  ObjDelayed        INTEGER,
  UtilDelayed       INTEGER,
  RuleType          INTEGER,
  GroupDelayed      INTEGER,
  FlexEligible      SMALLINT,
  MeterDeferred     INTEGER
)
...
;

```

### Syntax Elements

#### *RequestType*

Request type:

- O = Return the system query or system session delay queue
- W = Return the workload delay queue
- A = Return queries on the Delay and Defer queue.



- M = Return queries on the Defer queue only.

**Username**

Logon user name of session.

**HostId**

Host ID of the session number for the delayed request.

**SessionNo**

Session number for the held request.

**RequestNo**

Request number for the delayed request.

**RuleName**

Rule name identified in the Rule ID field.

**RuleId**

Rule ID for the workload or the system query or system session throttle rule that caused the query to be delayed or the rule ID for an arrival rate meter that caused the query to be deferred.

**TotalTimeHeld**

Total number of wall clock seconds that this request or session has been held.

**BlockingCnt**

Count of the number of consecutive times that this request has been identified as blocking at least one other session.

The value is zero if *Request Flag* is 6.

**OverRidable**

Request or session allowed to be aborted or released by the administrator. A session cannot be released if it exceeds the internal AMP worker task limit. A delayed session can always be aborted.

If the value is Y, the request or session is overridable.

If the value is N, the request or session is not overridable.

The queue table requests are controlled internally by the database and cannot be altered by the administrator.

If *Request Flag* is 6, this field indicates if the delayed session can be released. A session cannot be released if it exceeds the internal AMP worker task limit or an internal utility limit.

A delayed session can always be aborted.

#### **PEId**

PE VPROC ID which initiated this request.

#### **WDDelayed**

Indicator that the request is delayed for a workload rule.

A value of zero indicates the values is not delayed because of a workload throttle.

#### **ObjDelayed**

Indicator that the request is delayed for a system query or system session throttle rule.

A value of zero indicates the request is not delayed because of a system query or system session throttle.

#### **UtilDelayed**

Indicator that the request is delayed for a Utility rule.

A value of zero indicates the request is not delayed because of a utility throttle.

#### **RuleType**

Rule type for the request:

- 0 = Workload
- 1 = System
- 2 = Utility
- 3 = Workload Group
- 4 = Virtual Partition
- 5 = AWT Resource Limit
- 6 = Arrival Rate Meter

#### **GroupDelayed**

Whether Query is delayed for a workload group throttle: 1 (yes) or 0 (no).

This field is only valid on monitor version software ID 10 or later.

**FlexEligible**

The delayed request that is in a workload and is tagged as eligible for the Flex Throttle feature. Note, that flagged requests must meet all requirements for Flex Throttles before being released, such as system throttle restrictions.

- 0 = Flex eligible
- 1 = Not Flex eligible

**MeterDeferred**

Output parameter. It is the indicator of a request being deferred for an arrival rate meter.

- 0 = Query is not deferred
- 1 = Query is deferred

**Example: Using TDWMGetDelayedQueries to Get the System Query or System Session Delay Queue**

This example shows how to get the system query or system session delay queue.

```
SELECT * FROM TABLE (TDWM.TDWMGetDelayedQueries('O')) AS t1;
```

where 'O' is for system query or system session delay queue.

```
*** Query completed. One row found. 9 columns returned.
*** Total elapsed time was 1 second.
  Username  LUMBER
  HostId    1
  SessionNo 1010
  RequestNo 3
  WDName    test2
  WDIId     1
  TimeHeld  6676
  OverRidable Y
  BlockingCnt 0
```

**Example: Using TDWMGetDelayedQueries to Get the Workload Delay Queue**

This example shows how to get the workload delay queue.

```
SELECT * FROM TABLE (TDWM.TDWMGetDelayedQueries('W')) AS t1;
```

where 'W' is for workload delay queue.

```

*** Query completed. 2 rows found. 9 columns returned.
*** Total elapsed time was 1 second.
  Username LUMBER
    HostId 1
  SessionNo 1008
  RequestNo 3
    WDName test-wd
      WDIId 6
    TimeHeld 7131
OverRidable Y
BlockingCnt 0
  Username LUMBER
    HostId 1
  SessionNo 1009
  RequestNo 3
    WDName test-wd
      WDIId 6
    TimeHeld 7114
OverRidable Y
BlockingCnt 0

```

### Example: Using TDWMGetDelayedQueries to Get All Delay Queues

This example shows how to get all the delay queues.

```
SELECT * FROM TABLE (TDWM.TDWMGetDelayedQueries('A')) AS t1;
```

where 'A' or null is for all delay queues.

```

*** Query completed. 3 rows found. 9 columns returned.
*** Total elapsed time was 1 second.
Username HostId  SessionNo  RequestNo  WDName  WDIId  TimeHeld  OverRidable
-----
LUMBER    1      1010         3  test2    1      7717     Y
LUMBER    1      1008         3   test    6      7748     Y
LUMBER    1      1009         3   test    6      7731     Y

```

## TDWMGetDelayedUtilities

Returns the utility delay queue.

## Syntax

```

REPLACE FUNCTION TDWM.TDWMGetDelayedUtilities (
) RETURNS TABLE (
  Username      VARCHAR(128) CHARACTER SET UNICODE,
  HostId        SMALLINT,
  SessionNo     INTEGER,
  TotalTimeHeld INTEGER,
  OverRidable   CHAR CHARACTER SET LATIN,
  PEId          INTEGER,
  RequestNum    INTEGER,
  WDDelayed     INTEGER,
  ObjDelayed    INTEGER,
  UtilDelayed   INTEGER,
  RuleType      INTEGER
  RuleName      VARCHAR(128) CHARACTER SET UNICODE
)
...
;

```

## Syntax Elements

### Username

User name of the session.

### HostId

Host ID of the session number for the delayed utility.

### SessionNo

Session number for the held utility.

### TotalTimeHeld

Total number of wall clock seconds that this request has been held.

### OverRidable

Request or session allowed to be aborted or released by the administrator. A session cannot be released if it exceeds the internal AMP worker task limit. A delayed session can always be aborted.

If the value is Y, this request or session is overridable.

If the value is N, this request or session is not overridable.

The queue table requests are controlled internally by the database and cannot be altered by the administrator.

If *Request Flag* is 6, this field indicates if the delayed session can be released. A session cannot be released if it exceeds the internal AMP worker task limit or an internal utility limit.

A delayed session can always be aborted.

**RuleId**

ID of the rule that caused this query to be delayed.

**RequestNum**

Request number for the held utility.

**WDDelayed**

Indicator that the request is delayed for a workload rule.

A value of zero indicates the request is not delayed because of a workload throttle.

**ObjDelayed**

Indicator that the request is delayed for a system query or system session throttle rule.

A value of zero indicates the request is not delayed because of a system throttle.

**UtilDelayed**

Indicator that the request is delayed for a Utility rule.

A value of zero indicates the request is not delayed because of a utility throttle.

**RuleType**

Rule type for the request:

- 0 = Workload
- 1 = System
- 2 = Utility
- 3 = Workload group
- 4 = Virtual Partition

**RuleName**

Rule name identified in the Rule ID field.

**Example: Using TDWMGetDelayedUtilities**

```
SELECT * FROM TABLE (TDWM.TDWMGetDelayedUtilities()) AS t1;
*** Query completed. 4 rows found. 6 columns returned.
*** Total elapsed time was 1 second.
```

| Username | HostId | SessionNo | Total TimeHeld | OverRidable | RuleId |
|----------|--------|-----------|----------------|-------------|--------|
| USER1    | 1      | 2709      | 16             | 0           | 1      |
| USER1    | 1      | 2710      | 12             | 0           | 2      |
| USER1    | 1      | 2711      | 9              | 0           | 3      |
| DBC      | 1      | 2712      | 5              | 0           | 4      |

**TDWMInquire**

Returns the status of each category (that is, if it is active or not).

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMInquire (
) RETURNS TABLE (
    RuleSetId          INTEGER,
    Filter_category    VARCHAR(10) CHARACTER SET LATIN,
    Throttle_category  VARCHAR(10) CHARACTER SET LATIN,
    Workload_category  VARCHAR(10) CHARACTER SET LATIN,
    Vent_category      VARCHAR(10) CHARACTER SET LATIN
)
...
;
```

**Syntax Elements****RuleSetId**

ID of the rule set to apply. This must be the current rule set.

**Filter\_Category**

Current status of the Filters rule category. The value is either Active or Inactive.

**Throttle\_Category**

Current status of the Throttle rule category. The value is either Active or Inactive.

**Workload\_Category**

Current status of the WD rule category. The value is either Active or Inactive.

**Event\_Category**

Current status of the Event category. The value is either Active or Inactive.

**Example: Using TDWMInquire**

```
SELECT * FROM TABLE (TDWM.TDWMInquire()) AS t1;
*** Query completed. One row found. 5 columns returned.
*** Total elapsed time was 5 seconds.
```

| RuleSetId | Filter_Category | Throttle_Category | Workload_Category | Event_Category |
|-----------|-----------------|-------------------|-------------------|----------------|
| 9060      | Inactive        | Inactive          | Active            | Active         |

**TDWMListWDs**

Returns a list of the WDs.

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMListWDs (
  EnabledOnly VARCHAR(1) CHARACTER SET LATIN
) RETURNS TABLE (
  RuleSetId    INTEGER,
  EnabledFlag  VARCHAR(10) CHARACTER SET LATIN,
  WDIId       INTEGER,
  WDName      VARCHAR(128) CHARACTER SET UNICODE
)
...
;
```

**Syntax Elements*****EnabledOnly***

Enabled workload names.

The values, Y and N, return enabled WDs only.

**RuleSetId**

Rule set in which the WD is located.

**EnabledFlag**

Indicator that the WDs specified are enabled.



**WDId**

WD ID.

**WDName**

WD ID.

**WDName**

Name of the WD.

**Usage Notes**

This table function is only supported in Constant Mode.

The TDWMListWDs function provides similar functionality to a TDWM LIST WD request. For more information about this interface, see [TDWM LIST WD](#).

**Example: Using TDWMListWDs**

```
SELECT * FROM TABLE (TDWM.TDWMListWDs('Y')) AS t1;
*** Query completed. 40 rows found. 4 columns returned.
*** Total elapsed time was 2 seconds.
```

| RuleSetId | EnabledFlag | WDId | WDName |
|-----------|-------------|------|--------|
| 9889      | Enabled     | 201  | WD1    |
| 9889      | Enabled     | 202  | WD2    |
| 9889      | Enabled     | 203  | WD3    |
| 9889      | Enabled     | 204  | WD4    |
| 9889      | Enabled     | 205  | WD5    |
| 9889      | Enabled     | 206  | WD6    |
| 9889      | Enabled     | 207  | WD7    |
| 9889      | Enabled     | 208  | WD8    |
| 9889      | Enabled     | 209  | WD9    |
| 9889      | Enabled     | 210  | WD10   |

**TDWMLoadUtilStatistics**

Returns the statistics on the load utilities that are available in the system.

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMLoadUtilStatistics (
) RETURNS TABLE (
  UtilityType VARCHAR(10) CHARACTER SET LATIN,
  UtilityCount SMALLINT,
```

```

    UtilityLimit SMALLINT
  )
  ...
;

```

## Syntax Elements

### UtilityType

Type of utility:

- 0 = MultiLoad + FastLoad
- 1 = MultiLoad
- 2 = FastLoad
- 3 = FastExport
- 4 = ARC
- 5 = Standalone Mload
- 6 = Standalone FastLoad
- 7 = Standalone FastExport
- 8 = TPT update Op: MultiLoad
- 9 = TPT load Op: Fastload
- 10 = TPT export Op: FastExport
- 11 = JDBC MultiLoad
- 12 = JDBC FastLoad
- 13 = JDBC FastExport
- 14 = CSP Save Dump (FastLoad)
- 15 = DSA
- 16 = DSA Backup
- 17 = DSA Restore
- 18 = MLOADX

### UtilityCount

Number of utilities of this type that are active.

### UtilityLimit

Current limit on the number of utilities of this type.

---

### Note:

There is one record for every utility.

---

**Example: Using TDWMLoadUtilStatistics**

```
SELECT * FROM TABLE (TDWM.TDWMLoadUtilStatistics()) AS t1;
```

```
*** Query completed. 4 rows found. 3 columns returned.
```

```
*** Total elapsed time was 3 seconds.
```

| UtilityType | UtilityCount | UtilityLimit |
|-------------|--------------|--------------|
| -----       | -----        | -----        |
| MultiLoad   | 0            | 30           |
| FastLoad    | 0            | 30           |
| FastExport  | 0            | 60           |
| ARC         | 0            | 350          |
| .           |              |              |
| .           |              |              |
| .           |              |              |

**TDWMReleaseDelayedRequest**

Releases a request or utility session in the Teradata dynamic workload management software defer or delay queue.

This function returns a zero if it is successful.

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMReleaseDelayedRequest (
  HostId SMALLINT,
  SessionNo INTEGER,
  RequestNo INTEGER,
  WDIId INTEGER
) RETURNS INTEGER
...
;
```

**Syntax Elements*****HostId***

Host ID for the session.

***SessionNo***

Number of the session.

***RequestNo***

Request number of the task.

A value of zero indicates the utility session is released.

### **WDId**

WD ID for the request in the defer or delay queue being acted on.

## **Usage Notes**

The TDWMReleaseDelayedRequest function provides similar functionality to a TDWM DELAY REQUEST CHANGE request. For more information about this interface, see [TDWM DELAY REQUEST CHANGE](#).

When issuing this Teradata dynamic workload management software function, it must be qualified by the database name, TDWM (see examples below).

### **Example: Using TDWMReleaseDelayedRequest to Release a Delayed Request**

```
SELECT TDWM.TDWMReleaseDelayedRequest(HostId, SessionNo, RequestNo, 0)
FROM TABLE (TDWMGetDelayedQueries('0')) AS t1
WHERE SessionNo=4531;
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
TDWMReleaseDelayedRequest(HostId,SessionNo,RequestNo,0)
-----
0
```

### **Example: Using TDWMReleaseDelayedRequest to Release Multiple Delayed Requests**

```
SELECT TDWM.TDWMAbortDelayedRequest(HostId, SessionNo, RequestNo, 0)
FROM TABLE (TDWMGetDelayedQueries('0')) AS t1
WHERE t1.Username='TwmUser33';
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
TDWMAbortDelayedRequest(HostId,SessionNo,RequestNo,0)
-----
0
```

## **TDWMRuleControl**

Updates the EnabledFlag of the rule and the EnabledFlag in the base state for the rule in the TDWM database.

### **Syntax**

```
REPLACE PROCEDURE TDWM.TDWMRuleControl (
  IN RuleSetId    INTEGER,
  IN RuleName     TD_ANYTYPE,
```

```

    IN Operation    VARCHAR(1) CHARACTER SET LATIN,
    IN TDWMLock     VARCHAR(1) CHARACTER SET LATIN
  )
  ...
;

```

## Syntax Elements

### *RuleSetId*

ID of the rule set in which the rule is found.

### *RuleName*

Name of the rule specified.

### *Operation*

Request to enable or disable the rule:

- E = Enable the rule
- D = Disable the rule

### *TDWMLock*

Type of request performed on the TDWMLock table:

- W = Wait for the TDWMLock
- A = Abort if the TDWMLock table is locked
- S = Perform the update without locking the TDWMLock table

The TDWM places an exclusive lock on the TDWMLock table at startup.

## Usage Notes

The TDWMRuleControl procedure must be used with rules defined only with the base state as this is the only state value that the procedure changes.

### Example: Using TDWMRuleControl

```
CALL TDWM.TDWMRuleControl(1, 'throttle1', 'D', 'W');
```

## TDWMSetLimits

Updates the system query or system session throttle limits of a rule in the TDWM database.

## Syntax

```

REPLACE PROCEDURE TDWM.TDWMSetLimits (
  IN RuleSetId      INTEGER,
  IN RuleName       TD_ANYTYPE,
  IN TDWMLock       VARCHAR(1) CHARACTER SET LATIN,
  IN SessionLimit   INTEGER,
  IN QueryLimit     INTEGER,
  IN UtilityLimit   INTEGER,
  IN TimeUnit       VARCHAR(1) CHARACTER SET LATIN,
  IN QualifyTime    INTEGER
)
  ...
;

```

## Syntax Elements

### **RuleSetId**

ID of the rule set in which the rule is found.

### **RuleName**

Name of the rule specified.

### **TDWMLock**

Type of request performed on the TDWMLock table:

- W = Wait for the TDWMLock
- A = Abort if the TDWMLock table is locked
- S = Perform the update without locking the TDWMLock table

The TDWM places an exclusive lock on the TDWMLock table at startup.

### **SessionLimit**

Maximum number of concurrent sessions. NULL indicates that the field is not changed.

### **QueryLimit**

Maximum number of concurrent queries. NULL indicates that the field is not changed.

If the rule is an arrival rate meter, this value is used with TimeUnit and QualifyTime to describe an arrival rate limit. The limit is QueryLimit/TimeUnit for QualifyTime. For example, a maximum of 10 (requests)/minute for 120 seconds.

**UtilityLimit**

Maximum number of concurrent utilities. NULL indicates that the field is not changed.

**TimeUnit**

The time unit of the arrival rate meter:

- H = hour
- M = minute
- S = second

**QualifyTime**

Number of seconds that the arrival rate must continuously remain at the limit before new requests are held in the Defer queue.

**Example: Using TDWMSetLimits**

```
CALL TDWM.TDWMSetLimits(1, 'throttle1', 'A', NULL, 10, NULL);
```

## TDWMSummary

Returns the Teradata dynamic workload management software WD summary data fields.

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMSummary (
) RETURNS TABLE (
  WDIId INTEGER,
  Arrivals INTEGER,
  Completions INTEGER,
  MinRespTime FLOAT,
  MaxRespTime FLOAT,
  AvgRespTime FLOAT,
  MinCPUTime FLOAT,
  MaxCPUTime FLOAT,
  AvgCPUTime FLOAT,
  DelayedCnt INTEGER,
  AvgDelayTime FLOAT,
  ExceptionCnt INTEGER,
  MetSLGCnt INTEGER,
  ActiveQueryCnt INTEGER,
  ActiveDelayedCnt INTEGER,
  ErrorCnt INTEGER,
  AbortCnt INTEGER,
```

```

CollectionDate INTEGER,
CollectionTime FLOAT,
ExceptionAbCnt INTEGER,
ExceptionMvCnt INTEGER,
ExceptionCoCnt INTEGER,
IntervalDelayCnt INTEGER,
RejectedCount INTEGER,
MovedInCount INTEGER,
VirtualPartNum INTEGER,
AvgIOWaitTime FLOAT,
MaxIOWaitTime FLOAT,
AvgOtherWaitTime FLOAT,
MaxOtherWaitTime FLOAT,
AvgCPURunDelay FLOAT,
MaxCPURunDelay FLOAT,
AvgSeqRespTime FLOAT,
MaxSeqRespTime FLOAT,
AvgLogicalIO FLOAT,
MaxLogicalIO FLOAT,
AvgLogicalKBs FLOAT,
MaxLogicalKBs FLOAT,
AvgPhysicalIO FLOAT,
MaxPhysicalIO FLOAT,
AvgPhysicalKBs FLOAT,
MaxPhysicalKBs FLOAT,
ActiveThrottleBypass INTEGER,
FlexActive INTEGER,
FlexComplete INTEGER,
FlexArrivals INTEGER,
DeferCnt      INTEGER,
ActiveDeferCnt INTEGER,
AvgDeferTime  FLOAT
)
...
;

```

## Syntax Elements

### WDId

WD ID.



**Arrivals**

Number of queries that were classified into this WD by the Teradata dynamic workload management software.

**Completions**

Number of queries that completed in this WD in this dashboard or logging interval with:

- No error code or an error code of 3158
- No abort flag
- Either an AMP count of > 0 or a Parser CPU > 0

**MinRespTime**

Minimum response time in centiseconds of any query.

**MaxRespTime**

Maximum response time in centiseconds of any query.

**AvgRespTime**

Average response time in centiseconds for this workload based on the total response time of all completions divided by number of completions.

**MinCPUTime**

Minimum CPU in milliseconds of all the queries that completed in this workload in this dashboard or logging interval, where *CPU* is the minimum CPU used by any one AMP of the query plus the Parser CPU time used by that same query.

**MaxCPUTime**

Maximum CPU in milliseconds of all the queries that completed in this workload in this dashboard or logging interval, where *CPU* is the maximum CPU used by any one AMP of the query plus the Parser CPU time used by that same query.

**AvgCPUTime**

Running total in milliseconds of Parser CPU time plus the total AMP CPU of each query that completed in this workload divided by the number of completions of queries in this workload in this dashboard or logging interval.

**DelayedCnt**

Number of queries that are delayed in this workload.

**AvgDelayTime**

Average delay time in centiseconds of all the queries that are delayed in this workload.

**ExceptionCnt**

Number of queries with exceptions in this workload.

**MetSLGCnt**

Number of queries that met WD Response Time SLG requirements.

If the WD does not have Response Time SLG requirement, the query is still counted as met.

**ActiveQueryCnt**

Number of active queries in this workload.

**ActiveDelayCnt**

Number of queries currently delayed in this workload.

**ErrorCnt**

Number of queries that finished in this WD in this dashboard or logging interval with an error code > 0 but not 3158 or 3156.

**AbortCnt**

Number of queries that finished in this WD in this dashboard or logging interval with an error code of 0 or 3156 and an abort flag.

**CollectionDate**

Date of data collection.

**CollectionTime**

Time of data collection.

**ExceptionAbCnt**

Number of queries that aborted due to a TDWM exception.

**ExceptionMvCnt**

Number of queries that were moved into this WD due to an exception.

**ExceptionCoCnt**

Number of queries that hit an exception but continued in this WD.

**IntervalDelayCnt**

Number of queries that were delayed in this WD during this interval.

**RejectedCount**

Number of queries that were rejected by TDWM due to a filter or throttle rule violation.

**MovedInCount**

Number of queries that were moved into this WD either due to an exception action or manual move via PMPC.

**VirtualPartNum**

Virtual Partition number of the WD.

**AvgIOWaitTime**

Average duration of sleeps in milliseconds due to the I/O rate handling over the life of all requests completing in this WD during this interval.

**MaxIOWaitTime**

Maximum duration of sleeps in milliseconds due to the I/O rate handling of a request completing in this WD during this interval.

**AvgOtherWaitTime**

This column is reserved for future use.

**MaxOtherWaitTime**

This column is reserved for future use.

**AvgCPURunDelay**

Average wait time in milliseconds in the run queue of all the requests completing in this WD during this interval.

**MaxCPURunDelay**

Maximum wait time in milliseconds in the run queue of a request completing in this WD during this interval.

**AvgSeqRespTime**

Average of the sum of the response times of the steps (as if all the steps had been executed sequentially) of all the requests completing in this WD during this interval, in milliseconds.

**MaxSeqRespTime**

Maximum sum of the response times of the steps (as if all steps had been executed sequentially) of a request completing in this WD during this interval, in milliseconds.

**AvgLogicalIO**

Average count of logical I/Os for a request completing in this WD during this interval.

**MaxLogicalIO**

Maximum count of logical I/Os for a request completing in this WD during this interval.

**AvgLogicalKBs**

Average logical I/O usage in KB of all requests completing in this WD during this interval.

**MaxLogicalKBs**

Maximum logical I/O usage in KB for a request completing in this WD during this interval.

**AvgPhysicalIO**

Average count of physical I/Os of all requests completing in this WD during this interval.

**MaxPhysicalIO**

Maximum physical I/O usage of all requests completing in this WD during this interval.

**AvgPhysicalKBs**

Average physical I/O usage in KB of all requests completing in this WD during this interval.

**MaxPhysicalKBs**

Maximum physical I/O usage in KB for a request completing in this WD during this interval.

**ActiveThrottleBypass**

The number of queries reported in the Completions column that were allowed to run due to the ThrottleBypass ruleset attribute.

**FlexActive**

The number of queries that are currently active in the system that were released by the Flex Throttle feature.

**FlexComplete**

The number of queries that were released by the Flex Throttle feature that completed during this period.

**FlexArrivals**

The number of new requests that became active due to Flex Throttle feature.

**DeferCnt**

The number of queries in this WD that are deferred because of an arrival rate meter rule.

**ActiveDeferCnt**

Number of currently deferred queries in this WD.

**AvgDeferTime**

Average defer time due to an arrival rate meter rule, in centiseconds, for all queries assigned to this WD.

**Usage Notes**

The TDWMSummary function provides similar functionality to a TDWM Summary request. For more information about this interface, see [TDWM SUMMARY](#).

TDWMSummary returns information for all queries completed in the dashboard interval for the WD.

**Example: Using TDWMSummary**

```
SELECT WDIId, Arrivals, MetSLGCnt, ActiveReqs FROM TABLE (TDWM.TDWMSummary())
AS t2
WHERE arrivals <>0;
*** Query completed. 13 rows found. 4 columns returned.
*** Total elapsed time was 1 second.
```

| WDId | Arrivals | MetSLGCnt | ActiveReqs |
|------|----------|-----------|------------|
| 213  | 8        | 8         | 0          |
| 217  | 1        | 0         | 0          |
| 221  | 3        | 3         | 0          |
| 240  | 109      | 91        | 1          |

**TDWMSummaryRate**

Returns the collection rate (the duration of the collection period in seconds).

**Syntax**

```
REPLACE FUNCTION TDWM.TDWMSummaryRate (
) RETURNS SMALLINT
...
;
```

## Usage Notes

Use the SetSessionRate function to set the collection rate for updating session-level statistics within memory.

The TDWMSummaryRate function provides similar functionality to a TDWM Summary request. For more information about this interface, see [TDWM SUMMARY](#).

### Example: Using TDWMSummaryRate

```
SELECT TDWM.TDWMSummaryRate();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 3 seconds.
TDWMSummaryRate()
-----
60
```

## TDWMThrottleStatistics

Returns statistics for throttled database system queries or sessions, throttled workloads, or all throttles.

### Syntax

```
REPLACE FUNCTION TDWM.TDWMThrottleStatistics (
  RequestType VARCHAR(1) CHARACTER SET LATIN
) RETURNS TABLE (
  ObjectType      VARCHAR(17) CHARACTER SET LATIN,
  RuleId          INTEGER,
  RuleName        VARCHAR(128) CHARACTER SET UNICODE,
  ObjectName      VARCHAR(128) CHARACTER SET UNICODE,
  SubName         VARCHAR(128) CHARACTER SET UNICODE,
  Active          INTEGER,
  ThrottleLimit   INTEGER,
  Delayed         INTEGER,
  ThrottleType    VARCHAR(1) CHARACTER SET LATIN,
  FlexEligible    SMALLINT,
  TimeUnit        VARCHAR(1) CHARACTER SET LATIN,
  CurQualifyTime  INTEGER,
  ReqQualifyTime  INTEGER
)
...
;
```

## Syntax Elements

### *RequestType*

Request type:

- A = Return all throttle statistics.
- Q = Return statistics for throttled database system queries.
- S = Return statistics for throttled database system sessions.
- W = Return statistics for throttled workloads.

### *ObjectType*

Type of object for this statistic:

- User
- Account
- Account String
- Profile
- Client
- Network Address
- Application
- Database
- Table
- Macro
- Stored Procedure
- Workload
- Utility
- Collection
- View
- Queryband
- Function
- Method
- Virtual partition
- RESAWT (AWT Resource Limit)
- Arrival Rate Meter

### *RuleId*

When this field is a workload, it contains the workload ID.

**RuleName**

When this field is a workload, it contains the workload name.

**ObjectName**

For table, macro, stored procedure, view, function, and method object types, this is the name of the qualifying database.

For all other types, this is the name of the object.

For more information on these object types, see the ObjectTypes column.

**SubName**

Name of the table, macro, stored procedure, view, function, or method.

**Active**

Number of requests currently active for this object. This is, the number of active queries or sessions depending on the information requested in the input record.

If ObjectType is Arrival Rate Meter, this value is used with TimeUnit to describe the current rate. The current rate is Active/TimeUnit, such as 8 (requests)/minute.

**ThrottleLimit**

Current Teradata dynamic workload management software limit on the object.

This is the limit on the number of queries or sessions depending on the information requested in the input record.

A value of -1 indicates that no system query or session throttle limit is defined.

Active and delayed sessions are not valid when the ThrottleLimit value is -1.

If ObjectType is Arrival Rate Meter, this is the arrival rate limit.

**Delayed**

Number of requests on the delay queue at this time for this object.

If ObjectType is Arrival Rate Meter, this is the number of requests being deferred by the arrival rate meter.

**ThrottleType**

Type of throttle statistic. Possible values include:

- Q = Throttled database system query
- S = Throttled database system session
- W = Throttled workload



**FlexEligible**

The delayed request that is in a workload and is tagged as eligible for the Flex Throttle feature. Note, that flagged requests must meet all requirements for Flex Throttles before being released, such as system throttle restrictions.

- 0 = Flex eligible
- 1 = Not Flex eligible

**TimeUnit**

Time unit of the arrival rate meter:

- H = hour
- M = minute
- S = second

**CurQualifyTime**

Number of seconds in which the current arrival rate has continuously remained at the limit.

**ReqQualifyTime**

Number of seconds in which the current arrival rate must continuously remain at the limit before new requests are held in the Defer queue.

**Usage Notes**

The TDWMThrottleStatistics function provides similar functionality to a TDWM STATISTICS request. For more information, see [TDWM STATISTICS](#).

When issuing this Teradata dynamic workload management software function, it must be qualified by the database name, TDWM (see examples below).

**Example: Using TDWMThrottleStatistics to Request Query Throttle Statistics**

```
select * from table (TDWM.TDWMThrottleStatistics('Q')) AS t1;
*** Query completed. 4 rows found. 8 columns returned.
*** Total elapsed time was 1 second.
```

| ObjectType | RuleId | RuleName   | ObjectName | Subname   | Active | ThrottleLimit | Delayed |
|------------|--------|------------|------------|-----------|--------|---------------|---------|
| User       | 1      | test2      | LUMBER     |           | 2      | 2             | 1       |
| Use        | 2      | test4      | LUMBER     |           | 1      | 1             | 0       |
| Table      | 3      | table test | LUMBER     | ORDERS    | 1      | 1             | 0       |
| Macro      | 4      | macro test | LUMBER     | VENDMACRO | 0      | 1             | 1       |

**Example: Using TDWMThrottleStatistics to Request Session Throttle Statistics**

```
select * from table (TDWM.TDWMThrottleStatistics('S')) AS t1;
```

```
*** Query completed. 2 rows found. 8 columns returned.
```

```
*** Total elapsed time was 1 second.
```

| ObjectType | RuleId | RuleName | ObjectName | Subname | Active | ThrottleLimit | Delayed |
|------------|--------|----------|------------|---------|--------|---------------|---------|
| User       | 1      | test2    | LUMBER     |         | 2      | 4             | 0       |
| User       | 2      | test4    | LUMBER     |         | 2      | 2             | 0       |

**Example: Using TDWMThrottleStatistics to Request Workload Throttle Statistics**

```
select * from table (TDWM.TDWMThrottleStatistics('W')) AS t1;
```

```
*** Query completed. 2 rows found. 8 columns returned.
```

```
*** Total elapsed time was 1 second.
```

| ObjectType | RuleId | RuleName    | ObjectName | Subname | Active | ThrottleLimit | Delayed |
|------------|--------|-------------|------------|---------|--------|---------------|---------|
| Workload   | 0      |             |            |         | -1     | 0             | 1       |
| Workload   | 2      | WD-ConsoleR |            |         | 0      | -1            | 0       |
| Workload   | 3      | WD-ConsoleH |            |         | 0      | -1            | 0       |
| Workload   | 4      | WD-ConsoleM |            |         | 0      | -1            | 0       |
| Workload   | 5      | WD-ConsoleL |            |         | 0      | -1            | 0       |
| Workload   | 1      | WD-Default  |            |         | 0      | -1            | 0       |
| Workload   | 6      | test-wd     |            |         | 0      | 0             | 2       |

**Example: Using TDWMThrottleStatistics to Request All Throttle Statistics**

```
SELECT ObjectType(FORMAT 'x(10)'), rulename(FORMAT 'x(17)'),
       ObjectName(FORMAT 'x(13)'), active(FORMAT 'Z9'),
       throttlelimit as ThrLimit, delayed(FORMAT 'Z9'), throttletype as ThrType
FROM TABLE (TDWM.TDWMTHROTTLESTATISTICS('A')) AS t1
ORDER BY 1,2;
```

```
*** Query completed. 10 rows found. 7 columns returned.
```

```
*** Total elapsed time was 1 second.
```

| ObjectType | RuleName          | ObjectName   | Active | ThrLimit | Delayed | ThrType |
|------------|-------------------|--------------|--------|----------|---------|---------|
| Table      | t900throttlerule7 | T9TESTUSER   | 1      | 1        | 6       | Q       |
| User       | session919rule    | T900APIUSER4 | 4      | 5        | 0       | S       |
| Workload   | WD-Default        |              | 0      | -1       | 0       | W       |
| Workload   | WD1               |              | 0      | -1       | 0       | W       |
| Workload   | WD3               |              | 1      | 1        | 0       | W       |
| Workload   | WD345678901234567 |              | 1      | 1        | 3       | W       |
| Workload   | WD4               |              | 0      | 1        | 0       | W       |
| Workload   | WD5               |              | 0      | -1       | 0       | W       |

|          |     |   |    |   |   |
|----------|-----|---|----|---|---|
| Workload | WD6 | 0 | -1 | 0 | W |
| Workload | WD7 | 0 | -1 | 0 | W |

# Workload Management: Query Band APIs

A query band can be retrieved through two types of interfaces:

- [PM/API](#)
- [Open APIs \(SQL Interfaces\)](#)

Before using the query band APIs, you may want to familiarize yourself with the following topics:

- [PM/API](#)
- [Open APIs \(SQL Interfaces\)](#)
- [Query Band API Features](#)
- [Examples Using PM/API and Open APIs](#)

## PM/API

This section describes the MONITOR QUERYBAND interface that uses CLIV2 or the Teradata JDBC Driver. This interface is referred to as a PM/API.

For details on using the MONITOR QUERYBAND Teradata JDBC Driver request, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

[MONITOR QUERYBAND](#) is the only query band PM/API request currently available. This request provides similar functionality to the MonitorQueryBand function.

## MONITOR QUERYBAND

Returns the concatenated transaction, session, and profile query band for the specified session.

### Input Data

| Field Name        | Data Type            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>IndByte</i>    | BYTE                 | Indicator bits that specify which fields to treat as NULL if you are using indicator mode.<br>Each bit in the byte corresponds to one field in the input data.<br>If data is supplied for that field, set the bit to zero.<br>If the data for that field is NULL (that is, there is no data supplied for that field), set the bit to 1.<br><br><b>Note:</b><br>The <i>IndByte</i> field is only required if the CLIV2 request is submitted in indicator mode. |
| <i>mon_ver_id</i> | SMALLINT<br>NOT NULL | MONITOR software version ID. This can be version 6 or later.                                                                                                                                                                                                                                                                                                                                                                                                  |

| Field Name          | Data Type | Description                                                                                                                                                                                                                                                                                                                    |
|---------------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |           | For a general explanation of monitor version choices, see <a href="#">MONITOR VERSION</a> .                                                                                                                                                                                                                                    |
| <i>host_id</i>      | SMALLINT  | Logical ID of a host (or client) with sessions logged on. For example, a <i>hostid</i> of zero identifies internal sessions or system console sessions. <i>host_id</i> cannot exceed 1023.                                                                                                                                     |
| <i>SessionNo</i>    | INTEGER   | Number of the session. <i>session_no</i> combined with <i>host_id</i> represents a unique session ID.                                                                                                                                                                                                                          |
| <i>RunPEVprocNo</i> | SMALLINT  | PE vproc number where the session runs. This is typically obtained from the RunVprocNo data field of the MONITOR SESSION response (see <a href="#">Group I Data Fields and JDBC ResultSet Columns</a> ). If this field is specified with NULL or zero, all PEs will be searched for the session, causing significant overhead. |

## Monitor Privileges

To use this request, you must have the MONSESSION privilege as part of your default role or this privilege must be granted directly to you.

For more information on roles and privileges, see:

- *Teradata Vantage™ - Database Administration*, B035-1093
- *Teradata Vantage™ - Advanced SQL Engine Security Administration*, B035-1100
- *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>

## Usage Notes

If you encounter a problematic query that is using a large amount of system resources, use the associated query band to identify the source application issuing the request.

The MONITOR QUERYBAND request cannot be used on internal sessions or sessions that are logged onto the MONITOR partition. The query band is not stored for these types of sessions and is only stored for SQL partitions.

## CLIV2 Response Parcels

The MONITOR QUERYBAND request is treated internally as a one statement request that generates one response. The statement response returned from Vantage contains the following sequence of parcel types:

| Parcel Sequence | Parcel Flavor | Length (Bytes) | Comments/Key Parcel Body Fields                                                  |
|-----------------|---------------|----------------|----------------------------------------------------------------------------------|
| Success         | 8             | 18 to 273      | StatementNo = 1<br>ActivityCount = 1<br>ActivityType = 168 (PCLMONQUERYBANDSTMT) |

| Parcel Sequence | Parcel Flavor | Length (Bytes)                                                                                                 | Comments/Key Parcel Body Fields                                                                                              |
|-----------------|---------------|----------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| DataInfo        | 71            | 6 to 64100                                                                                                     | Optional: This parcel is present if request was IndicData parcel.                                                            |
| Record          | 10            | <ul style="list-style-type: none"> <li>5 to 64100(record mode)</li> <li>6 to 64100 (indicator mode)</li> </ul> | Depending on the request (Data or IndicData), data is in record or indicator mode. This record contains the query band text. |
| EndStatement    | 11            | 6                                                                                                              | StatementNo = 2-byte integer                                                                                                 |
| EndRequest      | 12            | 4                                                                                                              | None                                                                                                                         |

## Response

### Note:

The statement described below corresponds to a ResultSet returned by the Teradata JDBC Driver, and each of the fields correspond to a ResultSet column returned by the Teradata JDBC Driver. For more information on ResultSets, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

The Record parcel returns the following field:

| Field/Column Name | Data Type                      | Description                                                               |
|-------------------|--------------------------------|---------------------------------------------------------------------------|
| QueryBand         | VARCHAR (0 to 12304), NOT NULL | Concatenated query band string for the transaction, session, and profile. |

The QueryBand column returns the following concatenated transaction, session, and profile query band text:

=T> transaction query band =S> session query band =P> profile query band

| If there...                 | The text will contain...                                         |
|-----------------------------|------------------------------------------------------------------|
| is a transaction query band | =T> transaction query band                                       |
| is a session query band     | =S> session query band                                           |
| is a profile query band     | =P> profile query band                                           |
| are no query bands          | NULL. The return string is zero bytes or NULL in indicator mode. |

## Sample Input - CLv2 Request

The following example shows how the parcels for a MONITOR QUERYBAND request, built by CLv2, appear when sent to the Teradata server.

### Note:

In this example the size of the response buffer is set at the maximum (64,000 bytes), although you can set it to any size.

| Flavor |      | Length | Body                                |                    |
|--------|------|--------|-------------------------------------|--------------------|
| Num    | Name | Bytes  | Field                               | Value              |
| 0001   | Req  | 16     | Request                             | MONITOR QUERYBAND  |
| 0003   | Data | 12     | HostId<br>SessionNo<br>RunPEVprocNo | 1<br>1002<br>16383 |
| 0004   | Resp | 6      | BufferSize                          | 64000              |

## Sample Input - Teradata JDBC Driver Request

For an example of how the PM/API request, built in Java, appears when sent to the database server, see *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Sample Output

The following example shows a concatenated query band string that is returned for the current transaction and session.

```
=T> job=x1; =S> org=Finance;report=Fin123;
```

## Related Information

For information on setting the query band for a session or transaction, see SET QUERY\_BAND in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

For information on setting a profile query band, see CREATE PROFILE in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Open APIs (SQL Interfaces)

This section describes the SQL interfaces for retrieving a query band and parsing the query band string for name and value pairs. The SQL interfaces consist of UDFs and external stored procedures that you can invoke from any application.

Query band external stored procedures run on the PE and can retrieve query bands directly from memory. Query band functions run on the AMPs and require message passing to retrieve the query band. The external stored procedures provide better performance than query band functions.

---

**Note:**

When issuing query band UDFs, you do not need to fully qualify them by database name. They are automatically searched by the DBS in the SYSLIB database.

When calling query band external stored procedures, you must fully qualify them by the database name, SYSLIB.

For examples of query band functions and external stored procedures, see the following topics.

---

## GetQueryBand

Returns the concatenated query band string for the current transaction, session, and profile.

### Syntax

```
REPLACE FUNCTION SYSLIB.GetQueryBand (
) RETURNS VARCHAR(12304) CHARACTER SET UNICODE,
...
;
```

### Usage Notes

The GetQueryBand function can be used in a SELECT statement and an input parameter for a table function.

This function is created in the SYSLIB database by the DEM DIP script. EXECUTE privileges are granted to PUBLIC by default. For more information, see [Requirements for Using the API](#).

### Example: Using GetQueryBand

```
SELECT GetQueryBand();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
GetQueryBand()
-----
=S> a=1;b=2;c=3;d=4;
```

### Related Information

For information on setting the query band for a session or transaction, see SET QUERY\_BAND in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.



For information on setting a profile query band, see CREATE PROFILE in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## GetQueryBandPairs

Returns the name and value pairs in the query band.

### Syntax

GetQueryBandPairs are two overloaded table functions. One requires the query band string to be passed in; the other uses internal calls to retrieve the query band.

```
REPLACE FUNCTION SYSLIB.GetQueryBandPairs (
  SearchType SMALLINT
) RETURNS TABLE (
  QBName VARCHAR(129) CHARACTER SET UNICODE,
  QBValue VARCHAR(257) CHARACTER SET UNICODE
)
...
;
```

```
REPLACE FUNCTION SYSLIB.GetQueryBandPairs (
  QueryBandIn TD_ANYTYPE,
  SearchType SMALLINT
) RETURNS TABLE (
  QBName VARCHAR(129) CHARACTER SET UNICODE,
  QBValue VARCHAR(257) CHARACTER SET UNICODE
)
...
;
```

### Syntax Elements

#### SearchType

Search type:

- 0 = Return the name-value pairs in both transaction and session query bands are returned.

If the same name occurs in both the transaction and the session query bands, the value for the transaction query band is returned.

- 1 = Return only the name-value pairs in the transaction query band.
- 2 = Return only the name-value pairs in the session query band.
- 3 = Return only the name-value pairs in the profile query band.

**QueryBandIn**

Query band string.

**QBName**

Name of the query band.

**QBValue**

Value of the query band.

**Usage Notes**

Both functions are created in the SYSLIB database by the DEM DIP script. EXECUTE privileges are granted to PUBLIC by default. For more information, see [Requirements for Using the API](#).

**Example: Using GetQueryBandPairs**

The following example shows how to call the GetQueryBandPairs function.

```
SELECT * FROM TABLE(GetQueryBandPairs(0)) AS t1;
*** Query completed. 3 rows found. 2 columns returned.
*** Total elapsed time was 1 second.
```

| QBName | QBValue    |
|--------|------------|
| ORG    | FINANCE    |
| JOB    | ENDOFMONTH |
| JOBID  | 193858     |

**Example: Using GetQueryBandPairs with MonitorQueryBand as Input**

The following example shows how to call the GetQueryBandPairs function using the MonitorQueryBand function as input.

```
SELECT * FROM TABLE(GetQueryBandPairs(MonitorQueryBand(1, 103, 16383), 0))
AS t1;
*** Query completed. 3 rows found. 2 columns returned.
*** Total elapsed time was 1 second.
```

| QBName | QBValue    |
|--------|------------|
| ORG    | FINANCE    |
| JOB    | ENDOFMONTH |
| JOBID  | 193858     |

## GetQueryBandSP

Returns the concatenated query band for the current transaction and session.

### Syntax

```
REPLACE PROCEDURE SYSLIB.GetQueryBandSP (
  OUT queryband VARCHAR(12304) CHARACTER SET UNICODE
)
  ...
;
```

### Syntax Elements

#### *queryband*

The query band output parameter contains the concatenated transaction, session, and profile query bands.

### Usage Notes

The GetQueryBandSP procedure retrieves the query band directly from memory.

This procedure is created in the SYSLIB database by the DEM DIP script. EXECUTE privileges are granted to PUBLIC by default. For more information, see [Requirements for Using the API](#).

For information on setting the query band for a session or transaction, see SET QUERY\_BAND in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

### Example: Using GetQueryBandSP

```
CALL SYSLIB.GetQueryBandSP(qb);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
queryband
-----
=T> aa=123;bb=yxs;cc=mmmm; =S> a=1;b=2;c=3;d=4;
```

## GetQueryBandValue

Returns the value of the specified name in the current query band.

If no matches are found, the return value is an empty string or NULL.

## Syntax

The GetQueryBandValue functions are two overloaded functions. One requires the query band string to be passed in; the other uses internal calls to retrieve the query band.

```
REPLACE FUNCTION SYSLIB.GetQueryBandValue (
  SearchType SMALLINT,
  QBName      TD_ANYTYPE
) RETURNS VARCHAR (257) CHARACTER SET UNICODE
  ...
;
```

```
REPLACE FUNCTION SYSLIB.GetQueryBandValue (
  QueryBandIn TD_ANYTYPE,
  SearchType  SMALLINT,
  QBName      TD_ANYTYPE
) RETURNS VARCHAR (257) CHARACTER SET UNICODE
  ...
;
```

## Syntax Elements

### SearchType

Search type:

- 0 = Return the value of the first name-value pair where name is specified by the *QBName* input argument.  
If the query band contains the name-value pairs for the transaction and session, the function searches the transaction name-value pairs first.
- 1 = Search the transaction name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input parameter.
- 2 = Search the session name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input parameter.
- 3 = Search the profile name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input parameter.

### QBName

Name in the query band pair to search for.

### QueryBandIn

A query band string or NULL.

## Usage Notes

You can use these functions to query the DBC.DBQLogTbl table based on the names and values specified in the QueryBand column.

Both functions are created in the SYSLIB database by the DEM DIP script. EXECUTE privileges are granted to PUBLIC by default. For more information, see [Requirements for Using the API](#).

### Example: Using GetQueryBandValue to Query the QueryBand Column in QryLogV

This example shows you how to query the DBC.DBQLogTbl table for all Teradata brands specified in the QueryBand column.

```
SELECT t1.queryband(FORMAT 'X(50)') FROM
  dbc.qrylogv t1
WHERE GetQueryBandValue(t1.queryband, 0, 'brand') = 'teradata'
      AND username='qbuser' ORDER BY t1.queryid;
```

Result:

```
QueryBand
-----
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
```

### Example: Using GetQueryBandValue to Query DBQLogTbl (QryLogV)

This example shows how to query the DBC.DBQLogTbl table for all cat names specified in the QueryBand column.

```
SEL GetQueryBandValue(t1.queryband, 0, 'cat') FROM
  dbc.qrylogv t1 where
  GetQueryBandValue(t1.queryband, 0, 'CAT') IS NOT NULL
  AND username='qbuser' ORDER BY t1.queryid;
```

Result:

```
GetQueryBandValue(QueryBand,0,'cat')
-----
asta
asta
asta
asta
```

```

asta
asta
asta
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella

```

## GetQueryBandValueSP

Returns the value of the specified name in the current query band.

### Syntax

```

REPLACE PROCEDURE SYSLIB.GetQueryBandValueSP (
  IN SearchType SMALLINT,
  IN QBName VARCHAR (129) CHARACTER SET UNICODE,
  OUT QBValue VARCHAR(257) CHARACTER SET UNICODE
)
...
;

```

### Syntax Elements

#### *SearchType*

Search type:

- 0 = Return the value of the first name-value pair where name is specified by the *QBName* input argument.  
  
If the query band contains name-value pairs for the transaction and session, the function searches the transaction name-value pairs first.
- 1 = Search the transaction name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input argument.
- 2 = Search the session name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input argument.

- 3 = Search the profile name-value pairs in the query band and return the value that corresponds to the name specified by the QBName input argument.

**QBName**

Name in the query band pair to search for.

**QBValue**

Value of specified name.

**Example: Using GetQueryBandValueSP**

```
CALL SYSLIB.GetQueryBandValueSP(1,'aa',qbval);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
QBValue
-----
111
```

## MonitorQueryBand

Returns the concatenated query band string for the current transaction, session, and profile. If there is no concatenated query band, the function returns an empty string.

**Syntax**

```
REPLACE FUNCTION SYSLIB.MonitorQueryBand (
  HostId      SMALLINT,
  SessionNo   INTEGER,
  RunVprocNo  SMALLINT)
RETURNS VARCHAR(12304) CHARACTER SET UNICODE
)
...
;
```

**Syntax Elements****HostId**

Logical ID of a host.

**SessionNo**

Number of the session to get the query band for.

**RunVprocNo**

PE vproc number where the session runs. This is obtained from the RunVprocNo field of a MONITOR SESSION response (see [Group I Data Fields and JDBC ResultSet Columns](#)).

**Usage Notes**

The privileges to use this function are not granted by default. The database administrator must explicitly grant the privilege to users that require it.

The MonitorQueryBand function provides similar functionality to a MONITOR QUERYBAND request. For more information about this interface, see [MONITOR QUERYBAND](#).

**Example: Using MonitorQueryBand**

```
SELECT MonitorQueryBand (1, 1003, 16382);
* * * Query completed. One row found. One column returned.
* * * Total elapsed time was 5 seconds.
MonitorQueryBand (1, 1003, 16382)
-----
=T= job=x1; =S= org=Finance; report=Fin123;
```

**Related Information**

For information on setting the query band for a session or transaction, see SET QUERY\_BAND in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

For information on setting a profile query band, see CREATE PROFILE in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

**QueryBandReservedNames\_TBF**

Returns all the query band names and descriptions, including those dropped from a release.

**Syntax**

```
REPLACE FUNCTION SYSLIB.QueryBandReservedNames_TBF (
) RETURNS TABLE (
  Queryband_Name      VARCHAR(128) CHARACTER SET UNICODE,
  Release_Introduced  CHAR(5) CHARACTER SET LATIN,
  Release_Dropped     CHAR(5) CHARACTER SET LATIN,
  Category            CHAR(1) CHARACTER SET LATIN,
  [ Max_Len           INTEGER, ]
  Default_Value       VARCHAR(256) CHARACTER SET UNICODE,
  Description         VARCHAR(512) CHARACTER SET UNICODE,
  Queryband_Type      CHAR(1) CHARACTER SET LATIN
```



```

)
;

REPLACE VIEW SYSLIB.QueryBandReservedNames AS
  SELECT Queryband_Name, Category, Max_Len,
         Default_Value, Description, Queryband_Type
  FROM TABLE (QueryBandReservedNames_TBF()) AS t1
  WHERE Release_Dropped IS NULL
;

```

## Syntax Elements

### ***Queryband\_Name***

Name of the query band.

### ***Release\_Introduced***

Version number of when the query band name was introduced.

The version number is at the granularity of a minor release in standard format (that is, two digits for a major release number followed by a period and two digits for a minor release number). For example, '15.0' and '15.10'.

### ***Category***

Query band name category:

- T = Teradata restricted name (directive to the database)
- A = Application query band name (recommended for use by Teradata, Customer, and Partner applications)
- I = Teradata internal name (for internal Teradata use)

### ***Max\_Len***

[Optional] Maximum length of the value for the property.

Default: 256

### ***Default\_Value***

Default value of the property, which is NULL.

### ***Description***

Description of how the query band name is used and whether it is appropriate for a session or transaction query band, or both.

**Queryband\_Type**

Query band type:

- S = The session query band.
- T = The Transaction query band.
- B = The session and transaction query band.

**Usage Notes**

The QueryBandReservedNames view returns the current query band names only.

The QueryBandReservedNames\_TBF function and QueryBandReservedNames view are created in the SYSLIB database by the DEM DIP script. The privileges to use the QueryBandReservedNames\_TBF function and QueryBandReservedNames view are granted to PUBLIC.

**Example: Using QueryBandReservedNames\_TBF**

```
SELECT queryband_name(FORMAT 'X(30)'), release_introduced from table(queryband
reservednames_tbf()) as t1 order by 1;
```

```
*** Query completed. 28 rows found. 2 columns returned.
```

```
*** Total elapsed time was 3 seconds.
```

| queryband_name          | release_introduced |
|-------------------------|--------------------|
| -----                   | -----              |
| ACTION                  | 12.00              |
| APPLICATIONNAME         | 12.00              |
| BLOCKCOMPRESSION        | 14.10              |
| CLIENTUSER              | 12.00              |
| DEADLINE                | 12.00              |
| DESTINATION             | 12.00              |
| GROUP                   | 12.00              |
| IMPORTANCE              | 12.00              |
| JOBDEADLINE             | 12.00              |
| JOBID                   | 12.00              |
| JOBLIN                  | 12.00              |
| JOBSEQ                  | 12.00              |
| MAXQUERYTIME            | 12.00              |
| PROXYROLE               | 13.00              |
| PROXYUSER               | 13.00              |
| QUERYISSUETIME          | 12.00              |
| REDRIVE                 | 14.10              |
| SOURCE                  | 12.00              |
| STARTTIME               | 12.00              |
| TVSMIGRATION            | 13.10              |
| TVSTEMPERATURE          | 13.10              |
| TVSTEMPERATURE_FALLBACK | 14.10              |

|                              |       |
|------------------------------|-------|
| TVSTEMPERATURE_FALLBACKCLOBS | 14.10 |
| TVSTEMPERATURE_PRIMARY       | 14.10 |
| TVSTEMPERATURE_PRIMARYCLOBS  | 14.10 |
| UTILITYDATASIZE              | 13.10 |
| UTILITYNAME                  | 13.10 |
| VERSION                      | 12.00 |

# Workload Management: Embedded Services System APIs

The following describes the embedded services system functions that are used to retrieve data from the Priority Scheduler and query band name values.

Before using the embedded services system APIs, you may want to familiarize yourself with the following topics:

- [Open APIs \(SQL Interfaces\)](#)
- [Embedded Services System API Features](#)

## Open APIs (SQL Interfaces)

These SQL interfaces consist of UDFs that you can invoke from any application.

---

### Note:

When issuing embedded services system UDFs, you do not need to fully qualify them by the database name, TD\_SYSFNLIB.

---

For examples of embedded services system UDFs, see the following topics.

## GetPSFVersion

Returns the current type of Priority Scheduler, WORKLOADS.

### Syntax

```
REPLACE FUNCTION TD_SYSFNLIB.GetPSFVersion (  
  ) RETURNS VARCHAR (10) CHARACTER SET LATIN,  
  ...  
;
```

### Usage Notes

Priority Scheduler is managed by TASM, and is configured using the Teradata Viewpoint workload management portlets. For more information on those portlets, see *Teradata® Viewpoint User Guide*, B035-2206.

### Example: Using GetPSFVersion

This example shows the current Priority Scheduler is using workload definitions.

```
SELECT TD_SYSFNLIB.GetPSFVersion();
*** Query completed. One row found. One column returned.
*** Total elapsed time was 1 second.
getpsfversion() WORKLOADS
```

## GetQueryBandValueSF

Returns the value of the specified name in the current query band. If no matches are found, the return value is an empty string or NULL.

### Syntax

The GetQueryBandValueSF functions are two overloaded functions. One requires the query band string to be passed in; the other uses internal calls to retrieve the query band.

```
CREATE FUNCTION TD_SYSFNLIB.GetQueryBandValueSF (
  QueryBandIn TD_ANYTYPE,
  SearchType SMALLINT,
  QBName TD_ANYTYPE
) RETURNS VARCHAR(257) CHARACTER SET UNICODE
...
;
```

```
CREATE FUNCTION TD_SYSFNLIB.GetQueryBandValueSF (
  SearchType SMALLINT,
  QBName TD_ANYTYPE
) RETURNS VARCHAR(257) CHARACTER SET UNICODE
...
;
```

### Syntax Elements

#### *QueryBandIn*

A query band string or NULL.

#### *SearchType*

A numeric argument specifying which query band type to search for the name:

- 0 = Return the value of the first name-value pair where name is specified by the *QBName* input argument.

If the query band contains name-value pairs for the transaction and session, the function searches the transaction name-value pairs first.

- 1 = Search the transaction name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input argument.
- 2 = Search the session name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input argument.
- 3 = Search the profile name-value pairs in the query band and return the value that corresponds to the name specified by the *QBName* input parameter.

**QBName**

Name in the query band pair to search for. The function returns the value in the name-value pair. A query band name is a maximum of 128 characters.

**Usage Notes**

The `GetQueryBandValueSF` function provides similar functionality to the `GetQueryBandValue` function. For more information about this interface, see [GetQueryBandValue](#).

This function allows NULL input argument and return values.

You can use this function to query the `DBC.DBQLogTbl` table based on the names and values specified in the `QueryBand` column.

**Example: Using GetQueryBandValueSF to Query QryLogV**

This example shows you how to query the `DBC.DBQLogTbl` table for all Teradata brands specified in the `QueryBand` column.

```
SELECT t1.queryband(FORMAT 'X(50)') FROM
  dbc.dbqlogtbl t1
WHERE GetQueryBandValueSF(t1.queryband, 0, 'brand') = 'teradata'
      AND username='qbuser' ORDER BY t1.queryid;
```

Result:

```
QueryBand
-----
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
=T> game=monopoly;brand=teradata;
```

**Example: Using GetQueryBandValueSF to Query DBQLogTbl (QryLogV)**

This example shows how to query the `DBC.DBQLogTbl` table for all cat names specified in the `QueryBand` column.

```
SEL GetQueryBandValueSF(t1.queryband, 0, 'cat') FROM
  dbc.dbqlogtbl t1 where
    GetQueryBandValue(t1.queryband, 0, 'CAT') IS NOT NULL
    AND username='qbuser' ORDER BY t1.queryid;
```

Result:

```
GetQueryBandValueSF(QueryBand,0,'cat')
```

```
-----
asta
asta
asta
asta
asta
asta
asta
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
sabella
```

## TD\_get\_COD\_limits

Retrieves the maximum CPU and I/O Capacity on Demand (COD) values from the Priority Scheduler.

### Syntax

```
CREATE FUNCTION TD_SYSFNLIB.TD_get_COD_limits (
) RETURNS TABLE (
  CPULimit SMALLINT,
  IOLimit SMALLINT
)
...
;
```

## Syntax Elements

### CPULimit

The CPU COD maximum value.

If the WM COD mode is set and PM COD packages are installed, the CPULimit output parameter returns values less than 100%.

The PM COD package specifies the maximum allowable values.

If the PM COD mode is set or there are no PM COD packages installed, the CPULimit output parameter returns 100%.

### IOLimit

The I/O COD maximum value.

If the WM COD mode is set and PM COD packages are installed, this output parameter returns values determined by the installed PM COD package.

The PM COD package specifies the maximum allowable values.

The PM COD package specifies the maximum allowable values.

If the PM COD mode is set or there are no PM COD packages installed, the IOLimit output parameter returns 100%.

## Usage Notes

There are two COD modes:

| Mode   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PM COD | <p>Platform Metering Capacity on Demand (PM COD) sets resource limits on all activities on the node.</p> <p>PM CPU COD uses hardware throttling at the node level. PM I/O COD throttling is done at the disk device level.</p> <p><b>Note:</b></p> <p>Both PM I/O COD and WM I/O COD throttling are identical.</p>                                                                                                                                                                                                                                       |
| WM COD | <p>Workload Management Capacity on Demand (WM COD) allows resource limitations to be applied to CPU and I/O for all database activity on the node.</p> <p>WM CPU COD uses software throttling at the node level. WM I/O COD throttling is done at the disk device level.</p> <p><b>Note:</b></p> <p>Both PM I/O COD and WM I/O COD throttling are identical.</p> <p>WM COD Max is the maximum WM COD setting in WM COD mode. If there is a TDWM ruleset that exceeds WM COD Max, a message is logged and WM COD is overridden and set to WM COD Max.</p> |



WM COD mode is enabled on new installations. On upgrades, PM COD mode is enabled. The system can be switched to WM COD mode, which disables PM COD.

There are two types of PM COD packages:

- `teradata-pme-config-xxx`, where `xxx` is the COD level (for example, 62.5, 75, 87.5, and so on). This package specifies both the CPU and I/O limits.
- `teradata-pme-config-cpu-xxx`, where `xxx` is the COD level (for example, 62.5, 75, 87.5, and so on). This package specifies only CPU limits.

---

**Note:**

The CPU and I/O limits are the maximum values specified by the PM COD packages that cannot be exceeded.

---

The function return values vary according to the mode and the PM COD packages installed.

- PM COD mode always returns 100% for the `CPUlimit` and `IOLimit`.
- WM COD mode returns 100% for the `CPUlimit` and `IOLimit`, if there are no `teradata-pme-config` packages installed.
- WM COD mode returns less than 100% for `CPUlimit` and `IOLimit`, depending on the `teradata-pme-config-xxx` package installed.
- WM COD mode returns less than 100% for `CPUlimit`, depending on the `teradata-pme-config-cpu-xxx` package installed.

Within Viewpoint you can set the values lower than the WM COD maximum values specified by the PM COD packages in WM COD mode.

If you activate a Teradata dynamic workload management software ruleset that has CPU and I/O values greater than the COD maximum, the Priority Scheduler will replace the CPU and I/O values specified by the installed PM COD packages with the maximum allowable values and log a message in the messages log.

The maximum CPU and I/O COD values are global and not affected by planned environments (OpEnv). If you have multiple planned environments with different WM COD values, this will be honored, but will be limited by the COD max value.

Some PM COD packages, such as `teradata-pme-config-62.5` or `teradata-pme-config-cpu-87.5`, have tenth of a percent CPU limit granularity. For example, if the CPU maximum allowable value is 62.5%, the `TD_get_COD_limits` function will return 63%.

**Example: Using `TD_get_COD_limits`**

This example shows how to retrieve the CPU and I/O COD maximum values from the Priority Scheduler.

```
SELECT * FROM TABLE (TD_SYSFNLIB.TD_get_COD_Limits( ) ) As d;
CPUlimit      IOLimit
-----
75            75
```

### Related Information

- For more information about planned environments or how to set CPU and I/O COD values in the Teradata Viewpoint Workload Designer portlet, see *Teradata® Viewpoint User Guide*, B035-2206.
- For details about how to monitor WM COD and recommendations for applying it, see *Workload Management Capacity on Demand and Other Hard Limits Orange Book*, TDN0009761.
- For details about the Priority Scheduler, see *Priority Scheduler for Linux SLES 11 Orange Book*, 541-0008867.

# Workload Management: Ruleset APIs

The following discusses APIs for performing ruleset operations without using Viewpoint Workload Designer. More specifically:

- Create or replace a TASM rule.
- Delete a TASM rule.
- Manage a rule (for example, enable or disable).
- Activate a ruleset.

Currently, only system throttle and meter rules are supported.

The APIs are external stored procedures (XSPs).

An entry is logged in the DBC.TDWMEventLog table when one of the XSPs is successfully completed.

To create a system throttle or meter rules:

1. If you are creating a throttle or an arrival rate meter (ARM) for a specific ruleset, retrieve the desired ruleset or config name from the TDWM.Configurations table. Use 'ALLRULESETS' as the ruleset name to create the same throttle or ARM in all existing rulesets.
2. Call TDWM.TDWMCreateSystemThrottle (or TDWM.TDWMCreateArrivalRateMeter) to create a system throttle or an ARM without any qualification criteria in one or all existing rulesets.
3. Call TDWM.TDWMAddClassificationForRule to add a classification criterion for a system throttle or an ARM in one or all existing rulesets. This step is repeated for each additional classification criterion, if necessary.
4. [Optional] Call TDWM.TDWMAddClassificationForTarget to add a sub-criterion for a Target group classification in one or all existing rulesets. This step is repeated for each sub-criterion, if necessary.
5. Call TDWM.TDWMAddLimitForRuleState to add the default limit or a query limit for a specific state. This step can be repeated for each state. At least one of these calls must specify the default limit.
6. Call TDWM.TDWMManageRule to enable the new system throttle or ARM in one or all existing rulesets.
7. [Optional] Call the XSP TDWM.TDWMActivateRuleset to activate an updated ruleset with the new system throttle or ARM.

See [Examples](#).

---

## Note:

These APIs directly modify a ruleset or rulesets in the TDWM database so they are not immediately visible through Viewpoint Workload Designer. You need to copy the modified ruleset or rulesets from the Ready section to the Working section to make the changes visible.

---

## TDWMCreateSystemThrottle

Creates a system throttle without any qualification criteria in one or all existing rulesets.

### Syntax

```
REPLACE PROCEDURE TDWM.TDWMCreateSystemThrottle (
  IN RulesetName TD_ANYTYPE,
  IN ThrottleName TD_ANYTYPE,
  IN Description TD_ANYTYPE,
  IN Attributes VARCHAR(20) CHARACTER SET LATIN,
  IN ReplaceOption CHAR(1) CHARACTER SET LATIN
)
  ...
;
```

### Syntax Elements

#### **RulesetName**

Name of the ruleset in which the throttle will be created. Use 'ALLRULESETS' to create the throttle in all existing rulesets. Cannot be null and its length must be between 1 and 30.

#### **ThrottleName**

Name of the throttle to be created. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters, and so on. Length must be between 1 and 30.

#### **Description**

Description of the throttle. This can be null. Maximum length is 80.

#### **Attributes**

See *Teradata Vantage™ - Workload Management User Guide* for detailed descriptions of these attributes. Attributes can be combined.

- Throttle type. 'C', 'I', and 'M' are mutually exclusive.
  - 'C' = collective
  - 'I' = individual
  - 'M' = member
- Disabling override:
  - 'D' = disable manual release or abort

**ReplaceOption**

- 'Y' = Delete the specified throttle and recreate it. If the specified throttle does not exist, an error is returned.
- 'N' = Create a new throttle. If the specified throttle exists, an error is returned.

## TDWMCreateArrivalRateMeter

Creates an arrival rate meter without any qualification criteria in one or all existing rulesets.

**Syntax**

```
REPLACE PROCEDURE TDWM.TDWMCreateArrivalRateMeter (
  IN RulesetName TD_ANYTYPE,
  IN ArrivalRateName TD_ANYTYPE,
  IN Description TD_ANYTYPE,
  IN Attributes VARCHAR(20) CHARACTER SET LATIN,
  IN ReplaceOption CHAR(1) CHARACTER SET LATIN
)
...
;
```

**Syntax Elements****RulesetName**

Name of the ruleset in which the arrival rate meter will be created. Use 'ALLRULESETS' to create the meter in all existing rulesets. Cannot be null and its length must be between 1 and 30.

**ArrivalRateName**

Name of the arrival rate meter to be created. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters, and so on. The name cannot be null and its length must be between 1 and 30 characters.

**Description**

Description of the arrival rate meter.

**Attributes**

See *Teradata Vantage™ - Workload Management User Guide* for detailed descriptions of these attributes. Attributes can be combined.

- Throttle type:
  - 'C' = collective

- Disabling override:
  - 'D' = disable manual release or abort
- Logging Only (warning):
  - 'L' = Only log a warning instead of delay or reject

### ***ReplaceOption***

- 'Y' = Delete the specified arrival rate meter and recreate it. If the specified arrival rate meter does not exist, an error is returned.
- 'N' = Create a new arrival rate meter. If the specified arrival rate meter exists, an error is returned.

## **TDWMAddClassificationForRule**

Adds one classification criterion for a specific TASM rule in one or all existing rulesets. This step is repeated for each additional classification criterion, if necessary. This XPS is intended to support any TASM rule type (for example, throttle, arrival rate meter, workload, filter, and so on.). However, only system throttle and arrival rate meter are supported in this release.

### **Syntax**

```
REPLACE PROCEDURE TDWM.TDWMAddClassificationForRule (
  IN RulesetName TD_ANYTYPE,
  IN RuleName TD_ANYTYPE,
  IN Description TD_ANYTYPE,
  IN ClassificationType VARCHAR(20) CHARACTER SET LATIN,
  IN ClassificationValue TD_ANYTYPE,
  IN ClassificationOperator VARCHAR(6) CHARACTER SET LATIN,
  IN ReplaceOption CHAR(1) CHARACTER SET LATIN
)
...
;
```

### **Syntax Elements**

#### ***RulesetName***

Name of the ruleset that contains the RuleName. Use 'ALLRULESETS' to indicate all existing rulesets. Cannot be null and its length must be between 1 and 30.

**RuleName**

Name of the rule to add the criterion. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters. A combination of RulesetName and RuleName uniquely identifies a specific rule. If RulesetName is 'ALLRULESETS', the criterion is added to the same rule name in all existing rulesets. It cannot be null. The length must be between 1 and 30.

**Description**

Description of the criterion.

**ClassificationType**

This table describes possible values for ClassificationType:

| Value       | Description                      | Group                 |
|-------------|----------------------------------|-----------------------|
| USER        | User name                        | Request Source        |
| ACCT        | Account name                     | Request Source        |
| ACCTSTR     | Account string                   | Request Source        |
| PROFILE     | Profile                          | Request Source        |
| APPL        | Application name                 | Request Source        |
| CLIENTADDR  | Client IP address                | Request Source        |
| CLIENTID    | Client logon ID                  | Request Source        |
| DB          | Database                         | Target                |
| TABLE       | Table                            | Target                |
| VIEW        | View                             | Target                |
| MACRO       | Macro                            | Target                |
| SPROC       | Stored procedure                 | Target                |
| FUNCTION    | User-defined function            | Target                |
| METHOD      | User-defined method              | Target                |
| SERVER      | QueryGrid server                 | Target                |
| STMT        | Statement type                   | Query Characteristics |
| ALLAMP      | All AMP request                  | Query Characteristics |
| MSR         | Multi statement request          | Query Characteristics |
| MINSTEPROWS | Minimum estimated step row count | Query Characteristics |
| MAXSTEPROWS | Maximum estimated step row count | Query Characteristics |

| Value        | Description                             | Group                 |
|--------------|-----------------------------------------|-----------------------|
| MINFINALROWS | Minimum estimated final row count       | Query Characteristics |
| MAXFINALROWS | Maximum estimated final row count       | Query Characteristics |
| MINSTEPTIME  | Minimum estimated step processing time  | Query Characteristics |
| MAXSTEPTIME  | Maximum estimated step processing time  | Query Characteristics |
| MINTOTALTIME | Minimum estimated total processing time | Query Characteristics |
| MAXTOTALTIME | Maximum estimated total processing time | Query Characteristics |
| JOIN         | Join type                               | Query Characteristics |
| FTSCAN       | Full table scan                         | Query Characteristics |
| MEMORY       | Memory usage                            | Query Characteristics |
| IPE          | Incremental Planning and Execution      | Query Characteristics |
| QUERYBAND    | Query Band                              | Query Band            |

**ClassificationValue**

Value of the classification type to match for this criterion. A name can include the following wildcard characters:

- '\*' = Matches zero or more characters
- '?' = Matches one character

| Classification Type | Classification Value |
|---------------------|----------------------|
| USER                | User name            |
| ACCT                | Account name         |
| ACCTSTR             | Account string       |
| PROFILE             | Profile name         |
| APPL                | Application name     |
| CLIENTADDR          | Client IP address    |
| CLIENTID            | Client logon ID      |
| DB                  | Database name        |
| TABLE               | Table name           |
| VIEW                | View name            |
| MACRO               | Macro name           |



| Classification Type | Classification Value                                                                                                                                                                                                                                                                |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPROC               | Stored procedure name                                                                                                                                                                                                                                                               |
| FUNCTION            | User-defined function name                                                                                                                                                                                                                                                          |
| METHOD              | User-defined method name                                                                                                                                                                                                                                                            |
| SERVER              | QueryGrid server name                                                                                                                                                                                                                                                               |
| STMT                | <ul style="list-style-type: none"> <li>• D = DDL</li> <li>• M = DML</li> <li>• S = SELECT</li> <li>• C = COLLECT STATISTICS</li> </ul> Or a combination of the values above.<br>This ClassificationType can only be included, so ClassificationOperator must be set to 'I'.         |
| ALLAMP              | Not applicable                                                                                                                                                                                                                                                                      |
| MSR                 | Integer ( $\geq 2$ ) specifying minimum statement count                                                                                                                                                                                                                             |
| MINSTEPROWS         | Integer ( $\geq 1$ ) specifying minimum estimated step row count                                                                                                                                                                                                                    |
| MAXSTEPROWS         | Integer ( $\geq 1$ ) specifying maximum estimated step row count                                                                                                                                                                                                                    |
| MINFINALROWS        | Integer ( $\geq 1$ ) specifying minimum estimated final row count                                                                                                                                                                                                                   |
| MAXFINALROWS        | Integer ( $\geq 1$ ) specifying maximum estimated final row count                                                                                                                                                                                                                   |
| MINSTEPTIME         | Decimal ( $\geq 0$ ) specifying minimum estimated step processing time                                                                                                                                                                                                              |
| MAXSTEPTIME         | Decimal ( $\geq 1$ ) specifying maximum estimated step processing time                                                                                                                                                                                                              |
| MINTOTALTIME        | Decimal ( $\geq 1$ ) specifying minimum estimated total processing time                                                                                                                                                                                                             |
| MAXTOTALTIME        | Decimal ( $\geq 1$ ) specifying maximum estimated total processing time                                                                                                                                                                                                             |
| JOIN                | Only one of these values is allowed. <ul style="list-style-type: none"> <li>• N = no join</li> <li>• A = any join type</li> <li>• P = product join</li> <li>• Q = no product join</li> <li>• U = unconstrained product join</li> <li>• V = no unconstrained product join</li> </ul> |
| FTSCAN              | Not applicable                                                                                                                                                                                                                                                                      |
| MEMORY              | Only one of these values is allowed. <ul style="list-style-type: none"> <li>• I = increased</li> <li>• L = large</li> <li>• V = very large</li> </ul>                                                                                                                               |
| IPE                 | Not applicable                                                                                                                                                                                                                                                                      |

| Classification Type | Classification Value       |
|---------------------|----------------------------|
| QUERYBAND           | Query Band name-value pair |

### **ClassificationOperator**

This parameter specifies whether:

- The classification criterion is for inclusion or exclusion.
- For Profile and User inclusion criteria, the implicit AND operation can be overridden with an OR operation (see below).

A request is qualified for a rule (throttle, AMR, workload, and so on) if:

1. It satisfies all inclusion criteria of the rule, and
2. It satisfies no exclusion criteria of the rule

When multiple criteria are attached to a rule name, TASM performs implicit OR/AND operations among the criteria.

- For multiple criteria from the Request Source group:
  - Criteria of the same ClassificationType are joined together by the OR operator. For example, if the criteria of a throttle include USER=user1, USER=user2, then the throttle is applied to requests from user1 OR user2.
  - Criteria of the different ClassificationType are joined together by the AND operator. For example, if the criteria of a throttle include USER=user1, ACCT=finance, then the throttle is only applied to requests from user1 AND finance account. This operation can be overridden so that USER and PROFILE criteria are joined together by the OR operator.
- For multiple criteria from the Target group (except SERVER), they are joined together by the OR operator. For example, if the criteria of a throttle include DB=db1, TABLE=db3.tableX, then the throttle is applied to requests that reference db1 OR db3.tableX.
- For multiple criteria from different groups, they are joined together by the AND operator. For example, if criteria include:
  - USER: user1, user2, and user3
  - DB: db1, db2
  - TABLE: db3.tableX, db4.tableY

Then the criteria are interpreted as: (user1 OR user2 OR user3) AND (db1 OR db2 OR db3.tableX OR db4.tableY)

| Values | Descriptions                                    |
|--------|-------------------------------------------------|
| I      | Inclusion criterion (mutually exclusive with E) |

| Values | Descriptions                                                                                                                                                                     |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| E      | Inclusion criterion (mutually exclusive with I)                                                                                                                                  |
| O      | Join Profile and User criteria with an OR operator <ul style="list-style-type: none"> <li>Only valid with PROFILE or USER</li> <li>Can be combined with either I or E</li> </ul> |

**ReplaceOption**

- 'Y' = Delete the existing criteria of the specified rule and add the criterion. If the specified rule does not exist or does not have any criteria, an error is returned.
- 'N' = Add a criterion. If the specified criterion exists, an error is returned.

## TDWMAddClassificationForTarget

Adds a sub-criterion for a Target group classification (database, table, and so on) in one or all existing rulesets. The specified target must exist. This step is repeated for each additional sub-criterion, if necessary.

**Syntax**

```

REPLACE PROCEDURE TDWM.TDWMAddClassificationForTarget (
  IN RulesetName TD_ANYTYPE,
  IN RuleName TD_ANYTYPE,
  IN TargetType VARCHAR(20), CHARACTER SET LATIN,
  IN TargetValue TD_ANYTYPE,
  IN Description TD_ANYTYPE,
  IN ClassificationType VARCHAR(20) CHARACTER SET LATIN,
  IN ClassificationValue TD_ANYTYPE,
  IN ClassificationOperator VARCHAR(6) CHARACTER SET LATIN,
  IN ReplaceOption CHAR(1) CHARACTER SET LATIN
)
...
;

```

**Syntax Elements****RulesetName**

Name of the ruleset that contains the RuleName. Use 'ALLRULESETS' to indicate all existing rulesets. Cannot be null and its length must be between 1 and 30.

**RuleName**

Name of the rule to add the criterion. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters. A combination of RulesetName and RuleName uniquely identifies a specific rule. If RulesetName is 'ALLRULESETS', the criterion is added to the same rule name in all existing rulesets. The name cannot be null. The length must be between 1 and 30.

**TargetType and TargetValue**

These parameters identify the target to which the sub-criterion will be added. The maximum length of TargetValue is 256 characters. A TargetValue may include the following wildcard characters:

- '\*' = Matches zero or more characters
- '?' = Matches one character

This table describes possible values for TargetType and TargetValue:

| TargetType | TargetValue           |
|------------|-----------------------|
| DB         | Database name         |
| TABLE      | Table name            |
| VIEW       | View name             |
| MACRO      | Macro name            |
| SPROC      | Stored procedure name |

**Description**

Description of the criterion.

**ClassificationType**

Allowable values for ClassificationType depend on the specified TargetType. The following table describes possible values for ClassificationType for each TargetType.

|             | DB | TABLE | VIEW | MACRO | SPROC |
|-------------|----|-------|------|-------|-------|
| FTSCAN      | X  | X     | X    | X     | X     |
| JOIN        | X  | X     | X    | X     | X     |
| MINSTEPROWS | X  | X     | X    | X     | X     |
| MAXSTEPROWS | X  | X     | X    | X     | X     |
| MINSTEPTIME | X  | X     | X    | X     | X     |
| MINACCPCT   |    | X     |      |       |       |

|                                                | DB | TABLE | VIEW | MACRO | SPROC |
|------------------------------------------------|----|-------|------|-------|-------|
| STMT<br>• D (DDL)<br>• M (DML)<br>• S (SELECT) |    | X     |      |       |       |

MINACCPCT specifies the estimated minimum percentage of the specified table that a request accesses.

### ClassificationValue

This table describes possible values for ClassificationValue:

| Classification Type | Classification Value                                                                                                                                                                                                                                                                |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STMT                | <ul style="list-style-type: none"> <li>• D = DDL</li> <li>• M = DML</li> <li>• S = SELECT</li> </ul> Or a combination of the values above.                                                                                                                                          |
| MINSTEPROWS         | Integer ( $\geq 1$ ) specifying minimum estimated step row count                                                                                                                                                                                                                    |
| MAXSTEPROWS         | Integer ( $\geq 1$ ) specifying maximum estimated step row count                                                                                                                                                                                                                    |
| MINSTEPTIME         | Decimal ( $\geq 0$ ) specifying minimum estimated step processing time                                                                                                                                                                                                              |
| MAXSTEPTIME         | Decimal ( $\geq 0$ ) specifying maximum estimated step processing time                                                                                                                                                                                                              |
| JOIN                | Only one of these values is allowed. <ul style="list-style-type: none"> <li>• N = no join</li> <li>• A = any join type</li> <li>• P = product join</li> <li>• Q = no product join</li> <li>• U = unconstrained product join</li> <li>• V = no unconstrained product join</li> </ul> |
| FTSCAN              | Not applicable                                                                                                                                                                                                                                                                      |
| MINACCPCT           | Decimal ( $\leq 100$ ) specifying minimum percentage of table being accessed.                                                                                                                                                                                                       |

### ClassificationOperator

If ClassificationType value is FTSCAN, this parameter specifies whether the criterion is for inclusion or exclusion. For all other ClassificationType values of a target, the criterion must be set for inclusion.

- 'I' = criterion is for inclusion
- 'E' = criterion is for exclusion

'I' and 'E' are mutually exclusive.

### ***ReplaceOption***

- 'Y' = Delete the all existing sub-criteria of the specified target and add the sub-criterion. If the specified target does not exist or does not have any sub-criteria, an error is returned.
- 'N' = Add a sub-criterion. If the specified sub-criterion exists, an error is returned.

## **TDWMAddLimitForRuleState**

Adds the default limit or query limit for a specific state. This step is repeated for each state, if necessary. At least one of these calls must specify the default limit.

This XPS is intended to support any TASM rule types (for example, throttle, arrival rate meter, workload, filter, and so on). However, only system throttle and arrival rate meter are supported in this release.

### **Syntax**

```
REPLACE PROCEDURE TDWM.TDWMAddLimitForRuleState (
  IN RulesetName TD_ANYTYPE,
  IN RuleName TD_ANYTYPE,
  IN StateName TD_ANYTYPE,
  IN Description TD_ANYTYPE,
  IN StateLimit VARCHAR(50) CHARACTER SET LATIN,
  IN Action VARCHAR(6) CHARACTER SET LATIN,
  IN ReplaceOption CHAR(1) CHARACTER SET LATIN
)
...
;
```

### **Syntax Elements**

#### ***RulesetName***

Name of the ruleset that contains the RuleName. Use 'ALLRULESETS' to add limit for the specified rule state in all existing rulesets. Cannot be null and its length must be between 1 and 30.

#### ***RuleName***

Name of the throttle to add the limit. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters, and so on. Cannot be null and the length must be between 1 and 30.

**StateName**

Name of the state of the RuleName to add the limit. 'DEFAULT' is reserved for the default limit that is used for any state that does not have a specific limit. A combination of RulesetName, RuleName, and StateName uniquely identifies a specific rule state. If RulesetName is 'ALLRULESETS', the limit is added for the same StateName of the same RuleName in all existing rulesets.

**Description**

Description of the state limit. Description can be null. The maximum length is 80.

**StateLimit**

Depending on the rule type, the limit of a state may be specified in one of the following forms:

- 'NOLIMIT' = No limit for a state.
- '*number*' = A non-negative number specifying the limit of a state.
- '*arrivalRateLimit*' = An arrival rate meter limit. Use *arrivalRateLimit* only if the RuleName specifies an arrival rate meter. Use the format:

```
number/timeout QT=seconds
```

- *number*: A non-negative number specifying the limit of a state.
- *timeunit*: H = hour, M = minute, S = seconds
- *seconds*: A non-negative number specifying qualification time in seconds.

**Action**

Action to take when limit is exceeded. These choices are mutually exclusive.

- 'D' = delay
- 'R' = reject

**ReplaceOption**

- 'Y' = Delete the all existing limits of the specified state and add a new state limit. If the specified state does not exist or does not have any sub-criteria, an error is returned.
- 'N' = Add the specified state limit. If the specified state has a limit, an error is returned.

## TDWMMManageRule

Enables or disables a specific rule or rules. If the operation is to enable a rule, some validations are performed (for example, qualification criteria, default state limit) to reduce the risk of enabling an incomplete rule. If a rule fails any validation, the rule remains disabled.

## Syntax

```

REPLACE PROCEDURE TDWM.TDWMManageRule (
  IN RulesetName TD_ANYTYPE,
  IN RuleName TD_ANYTYPE,
  IN Operation CHAR(1) CHARACTER SET LATIN
)
  ...
;

```

## Syntax Elements

### *RulesetName*

Name of the ruleset that contains the RuleName. Use 'ALLRULESETS' to enable the specified rule in all existing rulesets. Cannot be null and its length must be between 1 and 30.

### *RuleName*

Name of the rule to be managed. A combination of RulesetName and RuleName uniquely identifies a specific throttle or an arrival rate meter. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters, and so on. Cannot be null and the length must be between 1 and 30.

### *Operation*

- 'E' = enable
- 'D' = disable

## TDWMActivateRuleset

Deletes a specific rule in one or all existing rulesets. It also deletes all classification criteria and state limits associated with the rule or rules.

## Syntax

```

REPLACE PROCEDURE TDWM.TDWMActivateRuleset (
  IN RulesetName TD_ANYTYPE
)
  ...
;

```



## Syntax Elements

### *RulesetName*

Name of the ruleset to be activated. Cannot be null and the length must be between 1 and 30.

## TDWMDeleteRule

Deletes a specific rule in one or all existing rulesets. It also deletes all classification criteria and state limits associated with the rule or rules.

## Syntax

```
REPLACE PROCEDURE TDWM.TDWMDeleteRule (
  IN RulesetName TD_ANYTYPE,
  IN RuleName TD_ANYTYPE
)
  ...
;
```

## Syntax Elements

### *RulesetName*

Name of the ruleset that contains the RuleName. Use 'ALLRULESETS' to delete the specified rule in all existing rulesets. Cannot be null and the length must be between 1 and 30.

### *RuleName*

Name of the rule to be deleted. The rule can be either a system throttle or an arrival rate meter. The specified name must be unique among all existing TASM rules: throttles, arrival rate meters, workloads, filters, and so on. Cannot be null and the length must be between 1 and 30.

## Examples

### Create System Throttle or ARM

In the MyFirstConfig ruleset, create a member system throttle called TableA\_FTS to set a concurrency limit of 1 with delay action for each user for requests that perform full table scans on the myDB.TableA table with estimated minimum total processing time of 3600 seconds from the WebApp application. This throttle is cannot be overridden.

The TASM rule attributes are:

- Ruleset name: MyFirstConfig
- System throttle name: TableA\_FTS

- Description: Member throttle FTS on myDB.TableA from WebApp
- Throttle type: Member
- Qualifications:
  - Application: WebApp
  - Table: myDB.TableA
  - Subcriteria on myDB.TableA: Full Table Scan and minimum total process time of 3600 seconds
- Limit: 1

The API call sequence:

1. The desired ruleset name, MyFirstConfig, is known so it is not necessary to query the TDWM.Configurations table.
2. Create a system throttle called TableA\_FTS without any classification criteria.

```
CALL TDWM.TDWMCreateSystemThrottle(
  'MyFirstConfig',      /* ruleset name */
  'TableA_FTS',         /* throttle name */
  'Member throttle FTS on myDB.TableA from WebApp' /* description */,
  'DM',                 /* 'D': disable override, 'M': member type */
  'N'                   /* not replace */);
```

3. Add classification criteria.

- Add application criterion:

```
CALL TDWM.TDWMAddClassificationForRule(
  'MyFirstConfig',      /* ruleset name */
  'TableA_FTS',         /* rule name */
  'Application classification', /* description */
  'APPL',               /* application criterion type */
  'WebApp',             /* APPL = WebApp */
  'I',                  /* Inclusion criterion APPL = WebApp */
  'N'                   /* Not a replace */);
```

- Add table type criterion:

```
CALL TDWM.TDWMAddClassificationForRule(
  'MyFirstConfig',      /* ruleset name */
  'TableA_FTS',         /* rule name */
  'Table classification', /* description */
  'TABLE',              /* table criterion type */
  'myDB.TableA',        /* TABLE = myDB.TableA */
  'I',                  /* Inclusion criterion */
  'N'                   /* Not a replace */);
```

4. Add sub-criteria for table criterion.

- Add full table scan sub-criterion for target myDB.TableA:

```
CALL TDWM.TDWMAddClassificationForTarget(
  'MyFirstConfig',          /* ruleset name */
  'TableA_FTS',            /* rule name */
  'TABLE',                 /* Target: TABLE criterion type */
  'myDB.TableA',           /* Target: TABLE = myDB.TableA */
  'FTSCAN sub-criterion',  /* description */
  'FTSCAN',               /* full table scan type */
  NULL,                   /* TargetValue not needed */
  'I',                    /* Inclusion criterion FTSCAN */
  'N'                     /* Not a replace */);
```

- Add minimum step time sub-criterion for target myDB.TableA:

```
CALL TDWM.TDWMAddClassificationForTarget(
  'MyFirstConfig',          /* ruleset name */
  'TableA_FTS',            /* rule name */
  'TABLE',                 /* Target: TABLE criterion type */
  'myDB.TableA',           /* Target: TABLE = myDB.TableA */
  'Min step time sub-criterion', /* description */
  'MINSTEPTIME',          /* minimum step time */
  '3600',                 /* min >= 3600 seconds */
  'I',                    /* Inclusion criterion */
  'N'                     /* Not a replace */);
```

5. Set the default limit for the system throttle:

```
CALL TDWM.TDWMAddLimitForRuleState(
  'MyFirstConfig',          /* ruleset name */
  'TableA_FTS',            /* rule name */
  'Default',               /* state name */
  'Default limit',         /* description */
  '1',                     /* limit */
  'D',                     /* delay */
  'N'                     /* Not a replace */);
```

6. Enable the system throttle:

```
CALL TDWM.TDWMManageRule(
  'MyFirstConfig',          /* ruleset name */
  'TableA_FTS',            /* rule name */
  'E'                      /* enable throttle */);
```

7. Call TDWM.TDWMActivateRuleset to activate the MyFirstConfig ruleset with the new throttle.

```
CALL TDWM.TDWMActivateRuleset(
  'MyFirstConfig'      /* ruleset name */);
```

### Delete System Throttle or ARM

1. If you want to delete a system throttle (or an ARM) in a specific ruleset, retrieve the desired ruleset name from the TDWM.Configurations table.
2. Call the XSP TDWM.TDWMDeleteRule to delete a system throttle (or an ARM) by specifying its name.

```
CALL TDWM.TDWMDeleteRule(
  'MyFirstConfig',      /* ruleset name */
  'Throttle_AcctDDL'    /* throttle name */);
```

3. Call the XSP TDWM.TDWMActivateRuleset to activate the updated MyFirstConfig ruleset, which no longer has the system throttle (or ARM).

```
CALL TDWM.TDWMActivateRuleset(
  'MyFirstConfig'      /* ruleset name */);
```

# Automated Statistics Management API

## Features and Examples

### Automating Statistics Collection and Monitoring

You can manage statistics collection manually using Automated Statistics Management open APIs or automatically using the Teradata Viewpoint Stats Manager portlet. For more information about this portlet, see the Teradata Viewpoint documentation.

To identify any already existing statistics that you want to incorporate into automated procedure:

1. Call an Automate-related open API.

This API copies the qualifying statistics definitions from the DBC dictionary to the TDSTATS database.

---

**Note:**

The TDSTATS database only stores metadata required to conduct Automated Statistic Management specific tasks on the statistics. You can use Teradata Studio or Teradata Studio Express to view details about the TDSTATS database, including tables, table columns, and views.

For a given database or table, the initial Automate operation is performed once.

If you subsequently collect additional statistics on these same tables or create new tables within a previously automated database, any resulting new dictionary stored statistics can be copied by calling the Automate operation again.

Teradata recommends that you periodically call the Automate operation on the same objects to ensure all statistics on those objects are automated. You can repeat the same Automate-related call as many times as needed.

- 
2. After a given set of statistics are automated, call the following open APIs:
    - a. PrepCollect to generate a prioritized list of collections based on the automated statistics definitions stored in the TDSTATS database. For more information, see [Preparing and Collecting Statistics](#) or [PrepCollect](#).
    - b. RunCollect or ReCollectTable to perform a recollection on the set of statistics. For more information, see [Preparing and Collecting Statistics](#).
    - c. DeAutomateStats to reverse the operation or the SelectAutomatedStats open API to list the statistics that are currently automated. For more information, see [DeAutomateStats](#) or [SelectAutomatedStats](#).

### Orphaned Statistics

If you issue an SQL DROP STATISTICS, DROP TABLE, or DROP DATABASE statement on a statistic that was previously automated, the resulting entry in the TDSTATS database is orphaned (that is, the

PrepCollect API will not prepare a collection on that particular statistic again). For more information about this open API, see [PrepCollect](#).

To remove an orphaned statistic from the TDSTATS database, you can call the ResyncStats open API. For more information about this open API, see [ResyncStats](#).

## Statistics from Dropped and Recreated Objects

A table is said to be Reincarnated after being temporarily dropped and subsequently recreated with the same column definitions, but potentially different data. Reincarnated tables are common to the ETL process where a "query" table is briefly dropped and then immediately recreated by renaming a "shadow" table that contains data from the original query table plus some recently loaded data. All reincarnated tables share the same underlying condition of having a newly assigned internal table id stored in their definition in system table DBC.TVM. Rather than leaving them orphaned and eventually removed, users have the option of repairing the TDStats stored definitions by marking them for preservation. For more information, see Open API [PreserveAfterRecreate](#).

## Example of Preserving TDStats Data During Table Recreation

A typical ETL flow involving a previously automated query table that is reincarnated by renaming a shadow table requires the following SQL statements and API calls.

Initial setup steps (performed once):

1. CREATE TABLE QueryTable (...);
2. Load or INSERT data into QueryTable
3. COLLECT STATISTICS .... ON QueryTable; /\* user defined stats on columns and indexes \*/
4. CALL AutomateStats('Db1','QueryTable',...);
5. CALL PreserveAfterRecreate('Db1','QueryTable',NULL,:NumStatsToPreserve);

---

### Note:

Step number 5 is required to ensure that TDStats metadata for QueryTable will survive the table recreation.

---

Table recreation steps (periodically repeated):

1. CREATE TABLE ShadowTable AS Db1.QueryTable WITH DATA AND STATS;
2. Load additional data into ShadowTable using bulk load methods
3. COLLECT SUMMARY STATISTICS ON ShadowTable; /\* refresh summary-only stats \*/
4. DROP TABLE QueryTable;
5. RENAME TABLE ShadowTable AS QueryTable;
6. CALL ResyncStats('Db1','QueryTable',:ResyncId,:NumRepaired);

**Note:**

Step number 6 is required to ensure that TDStats metadata for QueryTable will survive the table recreation.

**Excluded Statistics**

Automate-related open APIs automatically exclude statistics (such as Geospatial statistics) that do not support the SQL COLLECT STATISTICS statement with THRESHOLD option.

If a statistic is excluded, you will receive a warning message from the AutomateReport open API highlighting the statistic (for details, see [AutomateReport](#)).

For more information about the SQL COLLECT STATISTICS statement with THRESHOLD option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

**Functionality**

| If you want to ...                                                                                                                                       | Use the following SQL interface.<br>.. |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| copy statistics definitions for a specified database or table from the DBC dictionary to the TDSTATS database                                            | <a href="#">AutomateStats.</a>         |
| copy the definition of a single specified statistic from the dictionary to the TDSTATS database                                                          | <a href="#">AutomateSingleStat.</a>    |
| copy the definitions of those statistics identified by a prior Analyzer-related operation as actively used but not yet automated to the TDSTATS database | <a href="#">AutomateUsedStats.</a>     |
| remove specified statistics definitions from the TDSTATS database                                                                                        | <a href="#">DeAutomateStats.</a>       |
| list statistics definitions copied or removed by a prior invocation of an Automate-related open API                                                      | <a href="#">AutomateReport.</a>        |
| list automated statistics definitions that currently reside in the TDSTATS database                                                                      | <a href="#">SelectAutomatedStats.</a>  |
| remove orphaned statistics definitions from the TDSTATS database that are no longer defined in the dictionary                                            | <a href="#">ResyncStats.</a>           |

**Analyzing User Objects and Logged Query Plans**

You can use Analyzer open APIs to analyze user objects and logged query plans to identify conditions where future automated statistics management operations might be improved. These conditions include:

- Recommendation of additional statistics
- Identification of stale statistics that need recollected
- Identification of statistics that are not used by the Query Optimizer

When available, Analyzer open API results are leveraged by the PrepCollect open API in generating a prioritized list of SQL COLLECT STATISTICS statements.

Teradata recommends that you call Analyzer open APIs when there are major changes to:

- A query workload in which automated statistics are used.
- Table structures (for example, new indexes) on which automated statistics are defined.

By default, the status of each recommended new statistic from an Analyzer-related open API is unapproved which disqualifies it for inclusion by the PrepCollect and RunCollect open APIs. For more information on these open APIs, see [Preparing and Collecting Statistics](#).

After reviewing the recommendations from the Analyzer-related call (for example, AnalyzeStatsUsage open API), you can call the ApproveStat open API to approve one or more of the recommended statistics. Analyzer-related open APIs contain an input parameter that can mark newly recommended statistics approved and allow them to become eligible for inclusion by the PrepCollect open API.

## Functionality

| If you want to ...                                                                                                   | Use the following SQL interface ...       |
|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
| analyze statistics for a specified database or table using physical design information available from the dictionary | <a href="#">AnalyzeStats</a> .            |
| analyze statistics usage on specified objects referenced within the database query log                               | <a href="#">AnalyzeStatsUsage</a> .       |
| identify inactive statistics that have not been used during query optimization for a specified duration              | <a href="#">CleanupStats</a> .            |
| display results from prior call to the AnalyzeStats open API                                                         | <a href="#">AnalyzeStatsReport</a> .      |
| display results from prior call to the AnalyzeStatsUsage open API                                                    | <a href="#">AnalyzeStatsUsageReport</a> . |
| displays results from prior call to the CleanupStats open API                                                        | <a href="#">CleanupStatsReport</a> .      |
| approve a recommendation for collecting a new statistic                                                              | <a href="#">ApproveStat</a> .             |
| disapprove a recommendation for collecting a new statistic                                                           | <a href="#">DisapproveStat</a> .          |
| confirm a recommendation to inactivate a statistic                                                                   | <a href="#">InActivateStat</a> .          |
| reactivate a previously inactivated statistic                                                                        | <a href="#">ActivateStat</a> .            |

## Preparing and Collecting Statistics

You can use the set of Collect-related APIs to prepare and submit COLLECT STATISTICS commands on the automated statistic definitions stored in the TDSTATS database.

The collection process is divided into two phases:



- The prepare phase allows you to view and modify a prepared collection list prior to its actual submission. Each list of collection commands generated by a call to the PrepCollect open API is assigned a unique ID which is passed to the RunCollect open API to execute the list.
- The run phase allows you to specify a time limit. When the specified time expires, RunCollect will stop issuing any additional collections. If the specified time expires, you can resume the prior execution at a later time.

Although Teradata recommends that you call the PrepCollect open API before every call to RunCollect to incorporate the latest stale or missing statistics information, you can reuse the same list of collection commands in consecutive calls to RunCollect.

For more information, see [PrepCollect](#) or [RunCollect](#).

## Functionality

| If you want to ...                                                                                    | Use the following SQL interface ...           |
|-------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| prepare a prioritized list of COLLECT STATISTICS commands suitable for an execution or Run operation. | <a href="#">PrepCollect</a> .                 |
| execute the list of COLLECT STATISTICS commands generated by the PrepCollect open API                 | <a href="#">RunCollect</a> .                  |
| prepare and immediately execute collections on the automated statistics for a specified table         | <a href="#">ReCollectTable</a> .              |
| report a summarized list of pending, in-progress, and completed collections for an execution run      | <a href="#">RunCollectReport</a> .            |
| display the list of collection commands generated by the PrepCollect open API                         | <a href="#">SelectPreparedCollects</a> .      |
| display a list of completed collections for one or more past execution runs                           | <a href="#">SelectStatsExecutionHistory</a> . |
| abort pending or in-progress collections from an in-progress RunCollect                               | <a href="#">AbortCollect</a> .                |

## Controlling Recommended Actions and Overriding Default Settings

You can use the DBAControl open APIs to control the application of recommended actions by the Analyzer. The Analyzer mines metadata from DBQL, object use count, and the Data Definition Language processor to identify a critical subset of missing statistics identified by the Optimizer. If logged query plan data from DBQL is unavailable, physical design metadata (such as indexes, primary key or foreign key relationships, and so on) is used to identify missing statistics.

You can also use DBAControl open APIs to override any of the default settings stored in the TDSTATS database.

There are four sets of DBAControl open APIs:

- A set to exclude certain objects and their defined statistics from Automate, Analyzer, and Collect-related open APIs, where such exclusions supersede any inputs to those external stored procedures. For more information, see [DBAControl Open APIs for Excluding Objects](#).
- A set to control the various syntax options used in the preparing of SQL COLLECT STATISTICS statements. For more information, see [DBAControl Open APIs for Statistic Settings](#).
- A set to modify and customize the prepared list of collection commands. For more information, see [DBAControl Open APIs for Prepared Collections](#).
- A set to manage space in the TDSTATS database. For more information, see [DBAControl Open APIs for Managing Space](#).

## Functionality

### Using DBAControl for Excluding Objects

| If you want to ...                                                               | Use the following SQL interface ...        |
|----------------------------------------------------------------------------------|--------------------------------------------|
| exclude a database or table from all automated statistics management operations. | <a href="#">AddExcludedObject</a> .        |
| reverse a prior exclusion on specified objects                                   | <a href="#">RemoveExcludedObject</a> .     |
| show the current list of excluded objects                                        | <a href="#">SelectAllExcludedObjects</a> . |

### Using DBAControl for Statistics Settings

| If you want to ...                                                      | Use the following SQL interface ...           |
|-------------------------------------------------------------------------|-----------------------------------------------|
| specify histogram size for collections on specified objects             | <a href="#">UpdateStatHistogramSettings</a> . |
| specify sampling options for collections on specified objects           | <a href="#">UpdateStatSampleSetting</a> .     |
| specify threshold limits for refreshing statistics on specified objects | <a href="#">UpdateStatThresholdSetting</a> .  |

### Using DBAControl for Prepared Collections

| If you want to ...                                                      | Use the following SQL interface ...         |
|-------------------------------------------------------------------------|---------------------------------------------|
| add a specified collection to a prepared list                           | <a href="#">AddPreparedCollect</a> .        |
| remove a specified collection from the list                             | <a href="#">RemovePreparedCollect</a> .     |
| display the list of collections commands in a list                      | <a href="#">SelectPreparedCollects</a> .    |
| change the submission order of an individual collection within the list | <a href="#">PrioritizePreparedCollect</a> . |

## Using DBAControl for Managing Space

| If you want to ...                                                                       | Use the following SQL interface ...      |
|------------------------------------------------------------------------------------------|------------------------------------------|
| prevent the TDSTATS.AnalyzerHistoryTbl table from consuming an excessive amount of space | <a href="#">TruncateAnalyzerHistory.</a> |
| prevent the table from consuming an excessive amount of space                            | <a href="#">TruncateAutomateHistory.</a> |
| prevent the TDSTATS.CommandsHistoryTbl table from consuming an excessive amount of space | <a href="#">TruncateCollectHistory.</a>  |

## Other Advanced Settings

Other advanced settings intended for use by Teradata Services personnel or advanced users are not controlled by DBA Control APIs. Instead, they are stored as rows within a TDStats table and controlled by directly updating the rows with the SQL UPDATE statement.

Advanced settings are stored in TDStats.DefaultsTbl with the following default values:

| Setting Name                                | Default Value  | Description                                                                                                                                               |
|---------------------------------------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Max_Percent_Missing</i>                  | 20             | Max% of Analyzer Missing Recommendations to apply.                                                                                                        |
| <i>Min_Number_Missing</i>                   | 10             | Min# of Analyzer Missing Recommendations to apply.                                                                                                        |
| <i>Max_Threshold_Skip_Count</i>             | 2              | Maximum number of collection skips for Critical statistics.                                                                                               |
| Analyze_Logged_Usage_Events                 | Y              | Flag controlling analysis of logged usage of existing stats - System Default.<br>'Y' - perform function, 'N' - skip function                              |
| Analyze_Logged_Usage_Events_For_<JobName>   | Y              | Flag controlling analysis of logged usage of existing stats for individual Viewpoint Stats Manager job.                                                   |
| Analyze_Logged_Missing_Events               | Y              | Flag controlling analysis of logged missing stats recommendations - System Default.<br>'Y' - perform function, 'N' - skip function                        |
| Analyze_Logged_Missing_Events_For_<JobName> | Y              | Flag controlling analysis of logged missing stats recommendations for individual Viewpoint Stats Manager job.                                             |
| Analyze_Staleness                           | Y              | Flag controlling analysis of potentially stale statistics - System Default.<br>'Y' - perform function, 'N' - skip function                                |
| Analyze_Staleness_For_<JobName>             | Y              | Flag controlling analysis of potentially stale statistics for individual Viewpoint Stats Manager job.                                                     |
| Validate_Missing_Stats_Recommendations      | 1              | Validate Optimizer Missing Stats Recommendations within Analyze Jobs:<br>• 0 - Off<br>• 1 - On (default)                                                  |
| Validate_Missing_Stats_NonFatal_Error_Codes | '03523, 03524' | List of DBS error codes that do not fail Validation after EXPLAIN COLLECT STATISTICS failure. See related setting Validate_Missing_Stats_Recommendations. |
| Collect_Job_NonRetry_Error_Check            | 1              | Check if error is one that Collect Jobs should not retry:<br>• 0 - Off<br>• 1 - On (default)                                                              |
| Collect_Job_NonRetry_Error_Codes            | '02665, 02666, | List of DBS error codes that Collect Jobs should not retry after COLLECT STATISTICS error. See the related setting Collect_Job_NonRetry_Error_Check.      |

| Setting Name                             | Default Value | Description                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                          | 03598, 03807' |                                                                                                                                                                                                                                                                                                                                                                                              |
| Analyze_QueryLog_TimeLimit               | 0             | Maximum elapsed time in minutes for Analyze Jobs to spend analyzing logged queries. A zero or negative value indicates no limit. Applies to all Analyze Jobs that do not have an individual setting as specified by the related setting <i>Analyze_QueryLog_TimeLimit_For_&lt;JobName&gt;</i> .                                                                                              |
| Analyze_QueryLog_TimeLimit_For_<JobName> | 0             | Maximum elapsed time in minutes for the specified Viewpoint Stats Manager job to spend analyzing logged queries. Replace <JobName> with the actual job name.                                                                                                                                                                                                                                 |
| Monitor_Analyze_Job_Progress             | 0             | Enable progress monitoring of certain major functions performed by Analyze Jobs, including the analysis of logged queries for missing and used stats and the analysis of existing stats for potential staleness: <ul style="list-style-type: none"> <li>• 0 - Off (default)</li> <li>• 1 - On</li> </ul>                                                                                     |
| Analyze_Large_XML_Method                 | 1             | Controls how Analyze Jobs reads large XML text from the query log. The default is to use the UDF based method. <ul style="list-style-type: none"> <li>• 1 - UDF concat (default)</li> <li>• 2 - Client side concat</li> </ul>                                                                                                                                                                |
| Analyze_Parse_Parallel                   | Y             | Flag controlling use of parallelism for the parsing and processing of logged XML documents during analysis. Applies to all Analyze Jobs that do not have an individual setting as specified by the related setting <i>Analyze_Parse_Parallel_For_&lt;JobName&gt;</i> : <ul style="list-style-type: none"> <li>• 'Y' - Use parallelism (default)</li> <li>• 'N' - Skip parallelism</li> </ul> |
| Analyze_Parse_Parallel_For_<JobName>     | Y             | Flag controlling use of parallelism for the parsing and processing of logged XML documents during analysis for an individual Viewpoint Stats Manager job. <ul style="list-style-type: none"> <li>• 'Y' - Use parallelism (default)</li> <li>• 'N' - Skip parallelism</li> </ul>                                                                                                              |

To modify the values in this table use the SQL UPDATE statement.

### Example: Updating Advanced Settings

Change the system wide default setting for analyzing logged usage data. This example disables the Stats Manager Job function that analyzes logged data for existing stats usage.

```
UPDATE TDStats.DefaultsTbl
SET DefaultValue = 'N'
WHERE SettingName = 'Analyze_Logged_Usage_Events';
```

### Example: Changing Setting for Individual Stats Manager Job

Make the same setting change as shown in [Example: Updating Advanced Settings](#), but limit the scope of the change to an individual Stats Manager Job named MyAnalyzeJob.

```
INSERT INTO TDStats.DefaultsTbl (SettingName,DefaultValue)
VALUES ('Analyze_Logged_Usage_Events_For_MyAnalyzeJob','N');
```

### Example: Updating Settings for Analyze Jobs

Set the default time limit for all Analyze Jobs to 24 hours (1440 minutes).

```
UPDATE TDStats.DefaultsTbl
SET DefaultValue = '1440'
WHERE SettingName = 'Analyze_QueryLog_TimeLimit';
```

### Example: Updating Viewpoint Job Settings for Analyze Jobs

Set the time limit for Viewpoint Job “ANALYZE\_EIS\_DATABASES” to 3 hours (180 minutes).

```
INSERT INTO TDStats.DefaultsTbl (SettingName,DefaultValue)
VALUES ('Analyze_QueryLog_TimeLimit_For_ANALYZE_EIS_DATABASES','180');
```

### Example: Disabling Parallel Processing for Analyze Jobs

To disable parallel processing for all Analyze Jobs, use the UPDATE command:

```
UPDATE TDStats.DefaultsTbl
SET DefaultValue = 'N'
WHERE SettingName = 'Analyze_Parse_Parallel';
```

## Recommended Guidelines for Changing Advanced Settings

- If Viewpoint Stats manager Analyze Jobs take an excessively long time to complete, change the `Analyze_Logged_Usage_Events` setting to 'N'. Analyzing DBQL STATUSAGE requires extra processing overhead and the data obtained describes already existing statistics. Skipping this particular Analyze function will often improve job performance while still performing the more important function of analyzing missing statistics and recommending new statistics. To limit the scope of the changed setting to an individual job, insert a new row whose `SettingName` value is of the form `Analyze_Logged_Usage_Events_For_<JobName>` where `<JobName>` is replaced with the actual job's name.
- To increase the number of statistics recommendations reported by Analyze Jobs, increase the values for the settings `Max_Percent_Missing` and `Min_Number_Missing`. Analyzer logic is designed to identify and report a critical number of subset recommendations according to criteria that measure and rank their potential benefit. Increase these settings to see more of the candidate recommendations that would not ordinarily qualify.
- If the number of statistics recommendations reported by Analyze Jobs is overwhelming, decrease the values for the settings `Max_Percent_Missing` and `Min_Number_Missing`. If the quantity of Analyzer recommendations is excessive or the quality of many of them is poor, reducing these settings will further limit the recommendations to those candidates with the highest estimated benefits. The quality of a given recommendation is considered poor when its collection does not improve query performance.
- To ensure that statistics labeled critical by the Analyzer are forcibly recollected in spite of their defined thresholds, decrease the `Max_Threshold_Skip_Count` setting. Collect Jobs are capable of temporarily overriding the `THRESHOLD` for a critical statistic if its recollection has been deemed 'starved.' A statistic is first labeled 'critical' by the Analyzer if it is involved in steps with cardinality estimation errors. A

critical statistic becomes 'starved' if it has not been recollected in a configurable number of past Collect Job runs.

- The setting `Validate_Missing_Stats_Recommendations` should be left on (value of 1) to ensure that statistics recommendations from Analyze Jobs are valid and will not encounter parse-time failures when collected. Similarly, the setting `Collect_Job_NonRetry_Error_Check` should be left on (value of 1) to ensure that any statistics recommendations that encounter unexpected runtime failures during Collect Job runs are inactivated and not resubmitted in subsequent Collect Job runs. The settings `Validate_Missing_Stats_NonFatal_Error_Codes` and `Collect_Job_NonRetry_Error_Codes` can be used to further customize the validation and error checking behavior to define which DBS error codes are considered non-fatal (retryable) or fatal (non-retryable).
- `Analyze_QueryLog_TimeLimit` and `Analyze_QueryLog_TimeLimit_For_<JobName>` impose a time limit on that portion of Analyze Job processing that reads logged query data from the DBQL STATSUSAGE option. This phase of Analyze Jobs is typically the most time consuming, so it is the most relevant for imposing a time limit. Additional job time is required to aggregate and rank the logged query data and perform additional analysis functions that do not rely on query log STATSUSAGE data including staleness determination and the deactivation of unused statistics.
- Analyze Jobs whose elapsed times exceed the configured value in `Analyze_QueryLog_TimeLimit` or `Analyze_QueryLog_TimeLimit_For_<JobName>` will terminate successfully with generated missing stats recommendations based on the logged query data read and analyzed up to that point. A special event row representing time limit expiration is inserted into table `TDStats.AnalyzerHistoryTbl` with column `EventType` set to 'T' and column `EventDescription` reporting the number of queries analyzed before time expired along with the Viewpoint job name (up to first 20 characters).
- To impose a `TimeLimit` on a specific job only, users should insert a new row into `TDStats.DefaultsTbl` whose `SettingName` value is of the form `Analyze_QueryLog_TimeLimit_For_<JobName>` where substring `<JobName>` is replaced with the actual job's name as defined in Viewpoint Stats Manager. Additional rows can be inserted for each job that requires a custom time limit.
- When `Monitor_Analyze_Job_Progress` is enabled (on), Analyze Jobs will record special events that measure the progress of major analysis phases expressed as a percentage complete. Progress indicator events are inserted as special rows into table `TDStats.AnalyzerHistoryTbl` with column `EventType` set to 'P' and column `EventDescription` reporting the percentage of logged queries analyzed thus far or the percentage of existing stats whose staleness has been determined. Progress percentages and elapsed times are reset at the beginning of each major Analyze function phase. The relevant Viewpoint job name (up to first 20 characters) is included in the `EventDescription` column value.
- `Analyze_QueryLog_TimeLimit` and `Analyze_QueryLog_TimeLimit_For_<JobName>` can be used independently of `Monitor_Analyze_Job_Progress` or they can be used with `Monitor_Analyze_Job_Progress`. Progress can be monitored with or without specifying a time limit.
- Time limits specified by `Analyze_QueryLog_TimeLimit` and `Analyze_QueryLog_TimeLimit_For_<JobName>` should not be set so low as to prevent an Analyze Job from reading a representative subset of queries. Setting the time limit too low can result in Analyze Job recommendations that are not beneficial to large portions of the total query workload. Note that Analyze Jobs read queries from the log in ascending order of their DBQL assigned `QueryID` which means that older queries are generally read first.

- To skip a particular Analyze Job function, change the related settings `Analyze_Logged_Usage_Events`, `Analyze_Logged_Missing_Events`, or `Analyze_Staleness`. These settings provide a disabling mechanism for controlling the execution of Analyze Jobs functions and achieving the associated savings in elapsed time.
- Keeping the `Analyze_Parse_Parallel` setting at the system default of 'Y' has shown performance improvement in Analyze Jobs, but with an increase in spool usage. The extra spool is needed for the set-based amp-level parallel processing that is key to faster Analyze Jobs. If the increase in spool usage becomes problematic, the recommendations are to make more spool space available to the Analyze Job, or to reconfigure the Viewpoint Analyze Job to operate on a shorter period or portion of query log data. The other option is to change the `Analyze_Parse_Parallel` setting to 'N' to turn off parallelism.
- An increase in spool usage may also be seen with an `Analyze_Large_XML_Method` setting of '1'. Recommendations are to make more spool space available to the Analyze Job, reconfigure the Viewpoint Analyze Job to operate on a shorter period or portion of query log data, or to change the `Analyze_Large_XML_Method` setting to '2'.
- An `Analyze_Parse_Parallel` setting of 'Y' is incompatible with an `Analyze_Large_XML_Method` setting of '2' and will result in `Analyze_Parse_Parallel` being treated as if the setting had a value of 'N'.

## Reporting Progress Indicator and Timer Events

Analyzer events that report progress and time limit expiration can be retrieved from TDStats table `AnalyzerHistoryTbl`. The query below demonstrates how this is done for a Viewpoint Analyze Job named `Analyze_Personnel_DB`.

The output from the first execution of this query reports progress events for an Analyze job that is still in the process of reading query log rows. Output from the second execution of this same query at a later time reports progress events indicating that a time limit expired while reading query log rows and the Analyze Job has now transitioned to a subsequent phase that analyzes existing statistics for staleness. Output from the third execution of this query reports progress events indicating the job has finished analyzing all statistics for staleness.

Report the progress of the most recently submitted Analyze Job named `Analyze_Personnel_DB`:

```
SELECT Eventtimestamp (FORMAT 'YYYY-MM-DD hh:mi'), EventDescription
FROM TDStats.AnalyzerHistoryTbl
WHERE EventType IN ('T','P') AND EventDescription LIKE '%Job
Analyze_Personnel_DB%' AND
  AnalysisId = (SELECT MAX(AnalysisId) FROM TDStats.AnalyzerHistoryTbl WHERE
EventDescription LIKE '%Job Analyze_Personnel_DB%')
ORDER BY 1 ASC;
*** Query completed. 3 rows found. 2 columns returned.
*** Total elapsed time was 1 second.
EventTimeStamp  EventDescription
2015-04-2100:00  Progress of Job Analyze_Personnel_DBs: 0 of 500000 query log
rows analyzed (0 percent) after 1 mins
2015-04-2100:05  Progress of Job Analyze_Personnel_DBs: 5000 of 500000 query log
```



```
rows analyzed (1 percent) after 5 mins
2015-04-2100:11 Progress of Job Analyze_Personnel_DBs: 10000 of 500000 query
log rows analyzed (2 percent) after 11 mins
```

Execute the same SQL at a later time to report the updated progress of this same running job.

```
SELECT Eventtimestamp (FORMAT 'YYYY-MM-DD hh:mi'), EventDescription
FROM TDStats.AnalyzerHistoryTbl
WHERE EventType IN ('T','P') AND EventDescription LIKE '%Job
Analyze_Personnel_DBs%' AND
AnalysisId = (SELECT MAX(AnalysisId) FROM TDStats.AnalyzerHistoryTbl WHERE
EventDescription LIKE '%Job Analyze_Personnel_DBs%')
ORDER BY 1 ASC;
*** Query completed. 3 rows found. 2 columns returned.
*** Total elapsed time was 1 second.EventTimeStamp EventDescription
2015-04-2100:00 Progress of Job Analyze_Personnel_DBs: 0 of 500000 query log
rows analyzed (0 percent) after 1 mins
2015-04-2100:05 Progress of Job Analyze_Personnel_DBs: 5000 of 500000 query log
rows analyzed (1 percent) after 5 mins
2015-04-2100:11 Progress of Job Analyze_Personnel_DBs: 10000 of 500000 query
log rows analyzed (2 percent) after 11 mins
...
2015-04-2200:50 Progress of Job Analyze_Personnel_DBs: 110000 of 500000 query
log rows analyzed (22 percent) after 110 mins
2015-04-2200:55 Progress of Job Analyze_Personnel_DBs: 115000 of 500000 query
log rows analyzed (23 percent) after 115 mins
2015-04-2300:01 Time Expired for Job Analyze_Personnel_DBs: 120000 of 500000
query log rows analyzed with TimeLimit of 120 mins
2015-04-2300:02 Progress of Job Analyze_Personnel_DBs: 0 of 1000 stats analyzed
for staleness (0 percent) after 1 mins
2015-04-2300:02 Progress of Job Analyze_Personnel_DBs: 10 of 1000 stats
analyzed for staleness (1 percent) after 1 mins
2015-04-2300:02 Progress of Job Analyze_Personnel_DBs: 20 of 1000 stats
analyzed for staleness (2 percent) after 1 mins
```

Execute the same SQL at yet a later time to report the updated progress of this same running job.

```
SELECT Eventtimestamp (FORMAT 'YYYY-MM-DD hh:mi'), EventDescription
FROM TDStats.AnalyzerHistoryTbl
WHERE EventType IN ('T','P') AND EventDescription LIKE '%Job Analyze_Personnel
_DBs%' AND
AnalysisId = (SELECT MAX(AnalysisId) FROM TDStats.AnalyzerHistoryTbl WHERE E
ventDescription LIKE '%Job Analyze_Personnel_DBs%')
```



```

ORDER BY 1 ASC;
*** Query completed. 3 rows found. 2 columns returned.
*** Total elapsed time was 1 second.
EventTimeStamp  EventDescription
2015-04-2100:00  Progress of Job Analyze_Personnel_DBs: 0 of 500000 query log rows
                 analyzed (0 percent) after 1 mins
2015-04-2100:05  Progress of Job Analyze_Personnel_DBs: 5000 of 500000 query log r
ows analyzed (1 percent) after 5 mins
2015-04-2100:11  Progress of Job Analyze_Personnel_DBs: 10000 of 500000 query log
rows analyzed (2 percent) after 11 mins
...
2015-04-2200:50  Progress of Job Analyze_Personnel_DBs: 110000 of 500000 query log
rows analyzed (22 percent) after 110 mins
2015-04-2200:55  Progress of Job Analyze_Personnel_DBs: 115000 of 500000 query log
rows analyzed (23 percent) after 115 mins
2015-04-2300:01  Time Expired for Job Analyze_Personnel_DBs: 120000 of 500000 quer
y log rows analyzed with TimeLimit of 120 mins
2015-04-2300:02  Progress of Job Analyze_Personnel_DBs: 0 of 1000 stats analyzed f
or staleness (0 percent) after 1 mins
2015-04-2300:02  Progress of Job Analyze_Personnel_DBs: 10 of 1000 stats analyzed
for staleness (1 percent) after 1 mins
2015-04-2300:02  Progress of Job Analyze_Personnel_DBs: 20 of 1000 stats analyzed
for staleness (2 percent) after 1 mins
...
2015-04-2300:12  Progress of Job Analyze_Personnel_DBs: 980 of 1000 stats analyzed
for staleness (100 percent) after 11 mins
2015-04-2300:13  Progress of Job Analyze_Personnel_DBs: 1000 of 1000 stats analyze
d for staleness (100 percent) after 12 mins

```

## Creating Object Lists

There are some situations where the desired scope for an Automate, Analyze, or Collect API call involves a list of objects that cannot be specified by the *DatabaseName* and *TableName* or *ObjectName* input parameters even with the use of wildcard characters (see [AddQueryListEntry](#))

To solve this problem, you can create object lists that are capable of representing one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters that are dynamically resolved at the time of API invocation.

You can use these open APIs to create a named object list that represents any arbitrary set of database or tables. The named object list can be passed to an Automate, Analyzer, or a Collect-related open API.

**Functionality**

| If you want to ...                                                              | Use the following SQL interface ...    |
|---------------------------------------------------------------------------------|----------------------------------------|
| create a new object list with a user specified name                             | <a href="#">CreateObjectList</a> .     |
| add a database or table name to an existing named object list                   | <a href="#">AddObjectListEntry</a> .   |
| remove a list and the object entries within it                                  | <a href="#">RemoveObjectList</a> .     |
| display the database and table objects contained within a specified object list | <a href="#">SelectFromObjectList</a> . |

**Creating and Maintaining Query Lists**

A query list allows you to identify and analyze a set of queries based on arbitrary user-defined criteria.

Query List open APIs use one or more DBQL query IDs that are associated with a named identifier. This identifier can be passed to the Analyzer open API, [AnalyzeStatsUsage](#), as an input argument. For more information about this open API, see [AnalyzeStatsUsage](#).

You can use these open APIs to create and maintain query lists.

**Functionality**

| If you want to ...                    | Use the following SQL interface ...    |
|---------------------------------------|----------------------------------------|
| create a new query list               | <a href="#">AddQueryList</a> .         |
| add a query to a named list           | <a href="#">AddQueryListEntry</a> .    |
| drop a list and all queries within it | <a href="#">RemoveQueryList</a> .      |
| drop an individual query from a list  | <a href="#">RemoveQueryListEntry</a> . |

# Automated Statistics Management APIs

The following describes the different categories of the Automated Statistics Management open APIs in the form of SQL external stored procedures.

Before using the Automated Statistic Management external stored procedures, you must familiarize yourself with the following topics:

- [Automated Statistics Management API](#)
- [Automated Statistics Management API Features and Examples](#)
- [Granting Required Privileges](#)

## Granting Required Privileges

| You must have the following privilege as part of your default role or it must be granted directly to you... | To ...                                                                                                                                                                                                                                                                                    |
|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANT EXECUTE PROCEDURE ON TDSTATS TO <i>User</i>                                                           | call the Automated Statistic Management open APIs.                                                                                                                                                                                                                                        |
| GRANT SELECT ON TDSTATS TO <i>User</i>                                                                      | query the TDSTATS database directly.<br><b>Note:</b><br>If the Automated Statistics Management report open APIs (such as, AutomateReport, AnalyzeStatsReport, AnalyzeStatsUsageReport, and so on) are sufficient, you do not need to grant the SELECT privileges on the TDSTATS database. |
| GRANT EXECUTE ON DBC.dbqlaccessmacro TO <i>User</i>                                                         | enable statistics usage logging.                                                                                                                                                                                                                                                          |
| GRANT STATISTICS ON <i>User_Objects</i> TO TDSTATS                                                          | execute the AnalyzeStats or AnalyzeStatsUsage and RunCollect open APIs.                                                                                                                                                                                                                   |

## Related Information

| For information on ...                                                                    | See ...                                                                                                                                     |
|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| how to grant the EXECUTE PROCEDURE and SELECT privileges on the TDSTATS database          | <a href="#">GrantPrivileges.</a>                                                                                                            |
| how to grant the STATISTICS privilege on the qualifying user objects and TDSTATS database | <a href="#">RunCollect.</a>                                                                                                                 |
| the Automated Statistics Management report open APIs                                      | <ul style="list-style-type: none"> <li>• <a href="#">AnalyzeStatsReport.</a></li> <li>• <a href="#">AnalyzeStatsUsageReport.</a></li> </ul> |

| For information on ... | See ...                                                                                                                                                                         |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        | <ul style="list-style-type: none"> <li>• <a href="#">AutomateReport.</a></li> <li>• <a href="#">CleanupStatsReport.</a></li> <li>• <a href="#">RunCollectReport.</a></li> </ul> |

## GrantPrivileges

Grants the EXECUTE PROCEDURE and SELECT privileges on the TDSTATS database to the specified user.

### Syntax

```

REPLACE PROCEDURE TDSTATS.GrantPrivileges (
  IN UserName          VARCHAR(128) CHARACTER SET UNICODE,
  [ IN WithGrantOption CHAR(1) CHARACTER SET LATIN ]
)
...
;

```

### Syntax Elements

#### *UserName*

User name of the grantee.

#### *WithGrantOption*

[Optional] Indicator that the WITH GRANT OPTION privilege is granted. Possible values are Y or N.

Default: N

### Usage Notes

After the DIPSTATS script is run, user DBC (that is, the primary system user and owner or parent of all users, databases, and other objects) has all the privileges on the TDSTATS database, including the WITH GRANT OPTION privilege.

User DBC can call GrantPrivileges for users who need to call the TDSTATS external stored procedures and manually grant additional privileges (for example INSERT, DELETE, an UPDATE) on the TDSTATS database.

## Granting the STATISTICS Privilege

| To ...                                                                                                                                                                       | You should ...                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| call the following external stored procedures: <ul style="list-style-type: none"> <li>• RunCollect or RecollectTable</li> <li>• AnalyzeStats or AnalyzeStatsUsage</li> </ul> | Ensure you have permissions to grant the STATISTICS privilege. The STATISTICS privilege on user objects is not granted as part of GrantPrivileges.<br>Issue the following command on all user objects that have the SQL COLLECT STATISTICS statements submitted on them (where <i>user_object</i> is the database name): <pre>GRANT STATISTICS ON user_objects TO TDSTATS</pre> This command is performed only once for a given object. |
| reduce the number of individual privileges that must be maintained in the dictionary                                                                                         | perform the GRANT statement at the database level (that is, <i>user_object</i> ) whenever possible.                                                                                                                                                                                                                                                                                                                                     |

## Example: Using GrantPrivileges

The following example grants automated statistics management privileges to user PersonnelDBA.

```
.logon mysystem,dbc
CALL TDSTATS.GrantPrivileges ('PersonnelDBA','N');
*** Procedure has been executed.
*** Total elapsed time was 1 second
```

## Related Information

| For more information on ...                            | See ..                                                                                                                                                                                                             |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| the DIPSTATS script                                    | <ul style="list-style-type: none"> <li>• <a href="#">Open APIs (SQL Interfaces) Requirements for Using the API.</a></li> <li>• <i>Teradata Vantage™ - Database Utilities</i>, B035-1102.</li> </ul>                |
| the GRANT, WITH GRANT OPTION, or STATISTICS privileges | <ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - Advanced SQL Engine Security Administration</i>, B035-1100.</li> <li>• <i>Teradata Vantage™ - SQL Data Control Language</i>, B035-1149.</li> </ul> |
| calling RunCollect or RecollectTable                   | <ul style="list-style-type: none"> <li>• <a href="#">ReCollectTable.</a></li> <li>• <a href="#">RunCollect.</a></li> </ul>                                                                                         |

## Automate Open APIs

Use these open APIs (also called external stored procedures) to identify any already existing statistics that the you want to incorporate into automated procedures.

Automate-related external stored procedures do not support Geospatial statistics defined on columns with a complex data type. These statistics are excluded from the automation process. For statistics that cannot be automated, a warning message will be returned.

---

**Note:**

For more information about warning messages, see [AutomateReport](#).

---

## AutomateStats

Copies statistics definitions for a specified database or table from the dictionary to the TDSTATS database.

### Syntax

```
REPLACE PROCEDURE TDSTATS.AutomateStats (
  IN  DatabaseName      VARCHAR(128) CHARACTER SET UNICODE,
  IN  TableName         VARCHAR(128) CHARACTER SET UNICODE,
  IN  ObjectListName    VARCHAR(128) CHARACTER SET UNICODE,
  IN  NewerThan          TIMESTAMP(6) WITH TIME ZONE,
  IN  ExcludeTempTables CHAR(1) CHARACTER SET LATIN,
  IN  MarkApproved      CHAR(1) CHARACTER SET LATIN,
  [ IN DeleteOrphans     CHAR(1) CHARACTER SET LATIN, ]
  OUT AutomateId        BIGINT,
  OUT NumCopied         INTEGER,
  OUT NumRemoved       INTEGER
)
...
;
```

### Syntax Elements

#### ***DatabaseName***

Name of the database. *DatabaseName* limits the operation to statistics defined in the database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

#### ***TableName***

Name of the table. *TableName* limits the operation to statistics defined on the table you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

**ObjectListName**

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be automated.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

**NewerThan**

Time. *NewerThan* limits the operation to statistics that were initially collected after the specified time as recorded in the DBC.StatsTbl.CreateTimeStamp column.

**ExcludeTempTables**

Possible values:

- Y or NULL. If you specify Y or NULL, the statistics defined on global temporary tables of the specified databases are excluded from automation. The default value is Y.
- N. If you specify N, the statistics defined on global temporary tables of the specified databases are automated.

**MarkApproved**

Possible values:

- Y or NULL. If you specify Y or NULL, the copied statistic is marked approved and is included in subsequent calls to PrepCollect and RunCollect. The default value is Y.
- N. If you specify N, the copied statistic is marked unapproved. To approve the statistic, you can call ApproveStat.

**DeleteOrphans**

[Optional] Specify whether to remove previously automated statistics that no longer exist in the dictionary: Y (yes) or N (no).

Default: N

**AutomateId**

System supplied ID for the history results from this operation.

**NumCopied**

Copies or repairs statistics definitions for a specified database or table from the dictionary to the TDSTATS database. Repairs are limited to those objects that have been marked for preservation after dropping and recreating; see related API PreserveAfterRecreate.

**NumRemoved**

Number of statistics removed from the TDSTATS database.

**Usage Notes**

By default, the AutomateStats external stored procedure operates on statistics defined within all user databases defined in the system.

To automate never before collected statistics as defined by a list of SQL COLLECT STATISTICS statements residing in a text file, do the following:

- Submit the statistics to the database using BTEQ or another similar client tool.
- Call AutomateStats.

AutomateStats results are written to the TDSTATS.AutomateHistoryTbl table. To display the AutomateStats results, you can call AutomateReport.

**Wildcard Characters**

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

**Automating Statistics****Note:**

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To automate ...                                                                       | You must specify a value for ...                                                                                             |
|---------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| all statistics for a particular database                                              | <i>DatabaseName</i> .                                                                                                        |
| all statistics for a particular table                                                 | both <i>DatabaseName</i> and <i>TableName</i> .                                                                              |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures. |



| To automate ... | You must specify a value for ...                                                                                                                                                                                                              |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | <b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

**Example: Using AutomateStats**

The following example shows how to copy all statistics definitions for the Personnel database from the DBC dictionary to the TDSTATS database.

```
CALL TDSTATS.AutomateStats
('Personnel',NULL,NULL,NULL,'Y','Y','N',AutomateId,NumCopied,NumRemoved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
AutomateId  NumCopied  NumRemoved
-----
          1           5           0
```

**Related Information**

| For more information on ...            | See ..                                                                                                                            |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| creating or adding object list entries | <ul style="list-style-type: none"><li>• <a href="#">CreateObjectList.</a></li><li>• <a href="#">AddObjectListEntry.</a></li></ul> |
| displaying the AutomateStats results   | <a href="#">AutomateReport.</a>                                                                                                   |
| BTEQ                                   | <i>Basic Teradata® Query Reference</i> , B035-2414.                                                                               |

**AutomateSingleStat**

Copies the definition of a single specified statistic from the dictionary to the TDSTATS database.

**Syntax**

```
REPLACE PROCEDURE TDSTATS.AutomateSingleStat (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName    VARCHAR(128) CHARACTER SET UNICODE,
  IN StatsID      INTEGER,
  IN MarkedApproved CHAR(1) CHARACTER SET LATIN,
  OUT AutomateId  BIGINT,
```

```

    OUT NumCopied    INTEGER
)
...
;

```

## Syntax Elements

### ***DatabaseName***

Name of the database in which the statistic exists.

This input parameter cannot be NULL.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***TableName***

Name of the table on which the statistic exists.

This input parameter cannot be NULL.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***StatsID***

Dictionary ID for the statistic as stored in the DBC.StatsTbl.StatsId column.

This input parameter cannot be NULL.

### ***MarkedApproved***

Dictionary ID for the statistic as stored in the DBC.StatsTbl.StatsId column.

This input parameter cannot be NULL.

### ***AutomatId***

System supplied ID for the history results from this automation.

### ***NumCopied***

Possible values:

- 0 means the copy operation failed.
- 1 means the copy operation succeeded.

## Wildcard Characters

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

### Example: Using AutomateSingleStat

The following example shows how to automate the statistic defined on Personnel.Employee database whose dictionary assigned DBC.StatsTbl.StatsId column value is 1.

```
CALL TDSTATS.AutomateSingleStat ('Personnel', 'Employee', 1, 'Y',
AutomateId, NumCopied);
*** Procedure has been executed.
*** Total elapsed time was 10 seconds.
AutomateId    NumCopied
-----
          2          1
```

## AutomateUsedStats

Copies the definitions of those statistics identified by a prior Analyzer-related operation as having been used during query optimization but not yet automated to the TDSTATS database.

### Syntax

```
REPLACE PROCEDURE TDSTATS.AutomateUsedStats (
  IN  DatabaseName    VARCHAR(128) CHARACTER SET UNICODE,
  IN  TableName       VARCHAR(128) CHARACTER SET UNICODE,
  IN  ObjectListName  VARCHAR(128) CHARACTER SET UNICODE,
  IN  AnalysisID      INTEGER,
  IN  ExcludeTempTables CHAR(1) CHARACTER SET LATIN,
  IN  MarkApproved    CHAR(1) CHARACTER SET LATIN,
  OUT AutomateId      BIGINT,
  OUT NumCopied       INTEGER
)
```

```
...  
;
```

## Syntax Elements

### ***DatabaseName***

Name of the database. *DatabaseName* limits the operation to statistics defined on the database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***TableName***

Name of the table. *TableName* limits the operation to statistics defined on the table you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***ObjectListName***

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be automated.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***AnalysisID***

Result ID from a prior call to `AnalyzeStatsUsage`. For more information, see [AnalyzeStatsUsage](#).

This input parameter cannot be NULL.

### ***ExcludeTempTables***

Possible values:

- Y or NULL. If you specify Y or NULL, the statistics defined on global temporary tables of the specified databases are excluded from automation. The default value is Y.
- N. If you specify N, the statistics defined on global temporary tables of the specified databases are automated.

### ***MarkApproved***

Possible values:

- Y or NULL. If you specify Y or NULL, the statistics defined on global temporary tables of the specified databases are excluded from automation. The default value is Y.
- N. If you specify N, the copied statistic is marked unapproved. To approve the statistic, you can call ApproveStat.

**AutomateId**

System supplied ID for the history results from this operation.

**NumCopied**

Number of statistics copied to the TDSTATS database.

**Usage Notes**

The AutomateUsedStats external stored procedure identifies actively used statistics that must be automated and copies them.

The summarized results from this operation are written to the AutomateHistoryTbl table. To display these results, you can call AutomateReport.

Unless you specify a value of N in the ExcludeTempTables input parameter, any statistics defined on global temporary tables are automatically excluded under the assumption that their management must be highly customized within the user application that populates them.

**Wildcard Characters**

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

**Example: Using AutomateUsedStats**

This example shows how to perform the following operations:

- Analyze the statistics usage in the Personnel database for queries logged over a seven-day period.

```
CALL TDSTATS.AnalyzeStatsUsage ('2012-12-01 00:00:00.00','2012-12-31
00:00:00.00', NULL, 'Personnel', NULL, NULL, NULL, NULL, NULL, NULL,
NULL, 'Y', AnalysisId, NumEvents);
*** Procedure has been executed.
```

```
*** Total elapsed time was 10 seconds.
  AnalysisId  NumEvents
-----
          2          6
```

- Identify the results ID (for example, 2) from the call to AnalyzeStatsUsage as having been used during query optimization, if not already done.

```
CALL TDSTATS.AutomateUsedStats ('Personnel', NULL, NULL, 2, NULL, 'Y',
AutomateId, NumCopied);
*** Procedure has been executed.
*** Total elapsed time was 10 seconds.
  AutomateId  NumCopied
-----
          3          2
```

Related Information

| For more information on ...              | See ..                                                                                                                            |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| displaying the AutomateUsedStats results | <a href="#">AutomateReport.</a>                                                                                                   |
| creating or adding object list entries   | <ul style="list-style-type: none"><li>• <a href="#">CreateObjectList.</a></li><li>• <a href="#">AddObjectListEntry.</a></li></ul> |

DeAutomateStats

Reverses prior automate operations by removing specified statistics definitions from the TDSTATS database.

Syntax

```
REPLACE PROCEDURE TDSTATS.DeAutomateStats (  
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,  
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,  
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,  
  IN SCOID BIGINT,  
  IN InActiveSince TIMESTAMP(6) WITH TIME ZONE,  
  IN OrphansOnly CHAR(1) CHARACTER SET LATIN,  
  OUT DeAutomateId BIGINT,  
  OUT NumRemoved INTEGER)  
  ...  
;
```

## Syntax Elements

### **DatabaseName**

Name of the database from which all statistics should be de-automated.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### **TableName**

Name of the object list. *ObjectName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be de-automated.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### **ObjectName**

the object list. *ObjectName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be de-automated.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### **SCOID**

An individual statistic. *SCOID* limits the operation to an individual statistic as identified by its value as recorded in the TDSTATS.StatsTbl.SCOID column.

### **InActiveSince**

Time. *InActiveSince* limits the operation to those qualifying statistics that have been marked inactive by the CleanupStats or InActivateStat external stored procedure. For information about these external stored procedures, see [CleanupStats](#) and [InActivateStat](#).

### **OrphansOnly**

Possible values:

- Y. If you specify Y, the operation is limited to only those qualifying collections whose definitions no longer exist in the dictionary.
- N or NULL. If you specify N or NULL, all qualifying collections regardless of their existence in the dictionary are removed. The default value is N.

### **DeAutomateId**

Results ID for all events generated by this operation.

**NumRemoved**

Number of statistic definitions removed.

**Usage Notes**

If you specify all input parameters as NULL, all previously automated statistics for all objects in the system will be de-automated.

De-automated statistics, unless already orphaned, are not dropped from the dictionary and will remain accessible to query optimization. You must issue the SQL DROP STATISTICS statement to de-automate these statistics from the dictionary.

To display the DeAutomateStats results, you can call AutomateReport.

**Wildcard Characters**

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

**De-automating Statistics**

| To de-automate ...                                                                    | You must specify a value ...                                                                                                                                          |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single statistics collection                                                        | for <i>SCOID</i> .<br><b>Note:</b><br>When specifying a value for <i>SCOID</i> , you do not need to specify a value for the <i>DatabaseName</i> or <i>TableName</i> . |
| all statistics for a particular database                                              | for <i>DatabaseName</i> .                                                                                                                                             |
| all statistics for a particular table                                                 | for both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                   |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | for <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.                                      |



| To de-automate ...                                                                   | You must specify a value ...                                                                                                                                                                                                                  |
|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                      | <b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |
| only those qualifying statistics that have no recent usage in query optimization     | for <i>InactiveSince</i> .                                                                                                                                                                                                                    |
| only those qualifying statistics whose definitions no longer exist in the dictionary | of Y for <i>OrphansOnly</i> .                                                                                                                                                                                                                 |

### Example: Using DeAutomateStats

The following example shows how to de-automate all statistic definitions from the Personnel database.

#### Note:

The input argument value that begins with a colon represents a host variable whose value was populated in a prior call where it served as an output argument.

```
CALL TDSTATS.DeAutomateStats
('Personnel',NULL,NULL,NULL,NULL,'N',DeAutomateId,NumRemoved);
CALL TDSTATS.AutomateReport(:DeAutomateId);
```

### Related Information

| For more information on the ... | See ...                                                                                  |
|---------------------------------|------------------------------------------------------------------------------------------|
| SQL DROP STATISTICS statement   | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144. |
| DeAutomateStats results         | <a href="#">AutomateReport</a> .                                                         |

## AutomateReport

Returns a summary of events performed by a prior invocation of an Automate or Deautomate-related operation.

### Syntax

```
REPLACE PROCEDURE TDSTATS.AutomateReport (
  IN AutomateId BIGINT
)
```

```
...
;
```

## Syntax Elements

### *AutomateId*

Results ID from a prior AutomateStats or DeAutomateStats call. For more information, see [AutomateStats](#) or [DeAutomateStats](#).

This input parameter cannot be NULL.

## Output

| Column           | Description                                                                                                                                                                                                                                                                              |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DatabaseName     | Database from which statistics were automated.                                                                                                                                                                                                                                           |
| ObjectName       | Table from which statistics were automated.                                                                                                                                                                                                                                              |
| IndexNumber      | Dictionary ID of the index whose statistics were automated.<br><b>Note:</b><br>This value is NULL for non-indexes.                                                                                                                                                                       |
| IndexName        | User assigned name for the index, if any.                                                                                                                                                                                                                                                |
| ExpressionList   | List of comma separated fields on which statistic definitions are defined.                                                                                                                                                                                                               |
| EventType        | Event type: <ul style="list-style-type: none"> <li>• A means the event was automated successfully.</li> <li>• D means the event was deautomated.</li> <li>• W means a warning message was returned indicating that the operation on the statistics or object did not succeed.</li> </ul> |
| EventDescription | Text description of the event. For example, if the statistic cannot be automated, the EventDescription parameter returns a warning message.                                                                                                                                              |
| EventDate        | Time at which the Automation event occurred.                                                                                                                                                                                                                                             |
| Approved         | Flag indicates if the caller approved the copied statistics.                                                                                                                                                                                                                             |
| StatsType        | Type of statistics identified by the dictionary: <ul style="list-style-type: none"> <li>• B (Base table)</li> <li>• V (View)</li> <li>• Q (Query)</li> </ul> <b>Note:</b><br>V and Q statistic types are reserved for future use.                                                        |
| StatsId          | Dictionary assigned ID for the statistic as recorded in the DBC.StatsTbl.StatsId column.                                                                                                                                                                                                 |
| StatsName        | User assigned name for the statistic, if any.                                                                                                                                                                                                                                            |

| Column     | Description                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------------------|
| SCOID      | Automated Statistics Management assigned ID for the automated statistic as recorded in the TDSTATS.StatsTbl.SCOID column. |
| AutomateId | Automated Statistics Management assigned ID for the history results from this operation.                                  |

## Usage Notes

Report data is taken from the TDSTATS.AutomateHistoryTbl table along with corresponding data from the TDSTATS.StatsTbl and dictionary tables.

## Excluded Statistics

AutomateReport returns the following information for those statistics, such as Geospatial statistics, that are excluded from automation:

- A value of N in the Successful output column.
- A warning message in the EventDescription output column.

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Example: Using AutomateReport

The following example shows a summary of the actions performed by a prior call to the AutomateStats external stored procedure.

```
CALL TDSTATS.AutomateReport(1);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
*** ResultSet# 1 : 5 rows returned by "TDSTATS.AUTOMATEREPORT".
DatabaseName  ObjectName  IndexNumber  ExpressionList  EventDescription  .....
-----
PERSONNEL     EMPLOYEE    1            EmpNo          AUTOMATED STAT
PERSONNEL     EMPLOYEE    4            Name           AUTOMATED STAT
PERSONNEL     DEPARTMENT  1            DeptNo         AUTOMATED STAT
PERSONNEL     DEPARTMENT  ?            DeptName       AUTOMATED STAT
PERSONNEL     PROJECT     1            Proj_Id        AUTOMATED STAT
```

## SelectAutomatedStats

Displays the list of automated statistic definitions that are stored in the TDSTATS database.

### Syntax

```
REPLACE PROCEDURE TDSTATS.SelectAutomatedStats (
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE
)
...
;
```

### Syntax Elements

#### *DatabaseName*

Name of the database. *DatabaseName* limits the display to those statistics defined within the database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

#### *TableName*

Name of the table within the specified *DatabaseName*. *TableName* limits the display to those statistics for the table you specify within *DatabaseName*.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

#### *ObjectListName*

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be displayed.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### Output

| Column       | Description                                 |
|--------------|---------------------------------------------|
| DatabaseName | Database in which the statistic is defined. |
| TableName    | Table on which the statistic is defined.    |

| Column                  | Description                                                                                                                                                                                                                                                                         |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ExpressionList          | List of comma separated columns or expressions on which the statistic is defined.                                                                                                                                                                                                   |
| IndexNumber             | Dictionary assigned number of the index on which the statistic is defined as recorded in the DBC.StatsTbl.IndexNumber column.<br><b>Note:</b><br>This input parameter is NULL if the statistic is not collected on an index.                                                        |
| IndexName               | User assigned name for the index, if any.                                                                                                                                                                                                                                           |
| StatsId                 | Dictionary assigned ID for the statistic as recorded in the DBC.StatsTbl.StatsId column.                                                                                                                                                                                            |
| StatsName               | User assigned name for the statistic, if any.                                                                                                                                                                                                                                       |
| SCOID                   | TDSTATS database assigned ID for the automated statistic.                                                                                                                                                                                                                           |
| ApprovedStat            | Possible values:<br><ul style="list-style-type: none"> <li>• Y means the statistic is approved for collection.</li> <li>• N means the statistic is unapproved for collection.</li> <li>• D means the statistic is disapproved by the user.</li> </ul>                               |
| Excluded                | Possible values:<br><ul style="list-style-type: none"> <li>• Y means the statistic is defined on an object that has become excluded since the initial automation. For more information, see <a href="#">AddExcludedObject</a>.</li> <li>• N means the statistic is NULL.</li> </ul> |
| StatsState              | Possible values:<br><ul style="list-style-type: none"> <li>• M means the statistic is missing recommendations.</li> <li>• S means the statistic is stale.</li> <li>• C means the statistic is current.</li> </ul>                                                                   |
| StatsOrigin             | Possible values:<br><ul style="list-style-type: none"> <li>• U means the statistic is imported from the user by an Automate-related external stored procedure.</li> <li>• A means the statistic is automatically recommended by the system.</li> </ul>                              |
| StatsType               | Type of statistics identified by the dictionary:<br><ul style="list-style-type: none"> <li>• B (Base table)</li> <li>• V (View)</li> <li>• Q (Query)</li> </ul> <b>Note:</b><br>V and Q statistic types are reserved for future use.                                                |
| LastCollectTimeStamp    | Time of when the statistic was last collected.                                                                                                                                                                                                                                      |
| LastEvaluationTimeStamp | Time when the statistics was last submitted for collection by Automated Statistics Management. Depending on the SQL COLLECT STATISTICS THRESHOLD option in effect, the statistic may be recollected.                                                                                |

| Column               | Description                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | For more information about the THRESHOLD option, see COLLECT STATISTICS statement in <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144 or <i>Teradata Vantage™ - Data Dictionary</i> , B035-1092.                                                                                                                                                                                                 |
| SubmissionSkipCnt    | Number of consecutive runs where the statistic was not submitted for collection due to the time expiration or aborting.                                                                                                                                                                                                                                                                                                                 |
| ThresholdSkipCnt     | Number of consecutive submissions where the statistic did not meet its defined threshold for recollection.<br>For more information about the SQL COLLECT STATISTICS THRESHOLD option, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144 or <i>Teradata Vantage™ - Database Administration</i> , B035-1093.                                                                                    |
| UserImportance       | User assigned value for an entry that can be used to influence the priority of the statistic during the PrepCollect operation.<br>For more information about this external stored procedure, see <a href="#">PrepCollect</a> .                                                                                                                                                                                                          |
| SystemImportance     | System assigned value that is used in calculating the priority of the statistic during the PrepCollect operation.<br>For more information on this external stored procedure, see <a href="#">PrepCollect</a> .                                                                                                                                                                                                                          |
| ActiveStat           | Possible values: <ul style="list-style-type: none"> <li>• N means the statistic is marked inactive by the CleanupStats external stored procedure as a result of no observed usage by the Optimizer and is no longer incorporated by the PrepCollect external stored procedure.</li> <li>• Y means the statistic is actively used.</li> </ul>                                                                                            |
| LastAccessedByOpt    | Last recorded statistics usage by query optimizer according to system-wide activity counters.                                                                                                                                                                                                                                                                                                                                           |
| QryUsageFrequency    | Number of logged queries observed using the statistic.                                                                                                                                                                                                                                                                                                                                                                                  |
| MissingFrequency     | Number of queries observed wanting the missing statistic.                                                                                                                                                                                                                                                                                                                                                                               |
| EstPercentDataChange | Estimated percent of data changed since the last collection as determined by the most recent call to the Analyzer-related external stored procedure or PrepCollect.<br>For more information on this external stored procedure, see <a href="#">PrepCollect</a> .<br><br><b>Note:</b><br>The EstPercentDataChange value is NULL when the StatsState column value is M or when the EstPercentDataChange value is not a reliable estimate. |
| Age                  | Number of days since the last collected statistic. For more information, see the LastCollectTimeStamp output column.                                                                                                                                                                                                                                                                                                                    |
| Threshold            | Dictionary defined THRESHOLD setting for the statistic encoded as a text signature.                                                                                                                                                                                                                                                                                                                                                     |

| Column             | Description                                                                                                                                    |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | The Threshold column can be defined with the Age and EstPercentDataChange settings. For details, see the EstPercentDataChange and Age columns. |
| EstimateError      | Indicator that one or more queries observed have cardinality estimation inaccuracy involving the use or absence of this statistic.             |
| EstErrFrequency    | Number of queries observed with cardinality estimation inaccuracies.                                                                           |
| CreateUser         | User who originally automated the statistic.                                                                                                   |
| CreateTimeStamp    | Time in which the statistic was originally automated.                                                                                          |
| LastAlterUser      | User who last altered the TDSTATS database stored data for the statistic.                                                                      |
| LastAlterTimeStamp | Time in which the TDSTATS database data for this statistic was last altered.                                                                   |

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Wildcard Characters

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

## Displaying Statistics

### Note:

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To display ...                                                                        | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                             |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| all statistics for a particular database                                              | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                                        |
| all statistics for a particular table                                                 | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                              |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <p><i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |

## ResyncStats

Removes statistics definitions stored in the TDSTATS database that no longer exist in the dictionary as a result of a SQL DROP statement. In the case of objects marked for preservation after dropping and recreating, the statistics definitions are repaired rather than removed; see related API [PreserveAfterRecreate](#).

### Syntax

```

REPLACE PROCEDURE TDSTATS.ResyncStats (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName    VARCHAR(128) CHARACTER SET UNICODE,
  OUT Automateld  BIGINT,
  OUT NumRemoved INTEGER
)
...
;
```

### Syntax Elements

#### *DatabaseName*

Name of the database. *DatabaseName* limits the operation to statistics defined in the database you specify.

#### *TableName*

Name of the table. *TableName* limits the operation to the table you specify.



**AutomateId**

ID for the history results from this operation.

**NumRemoved**

Number of statistics removed from the TDSTATS database and the number of rows repaired.

**Usage Notes**

By default, the ResyncStats external stored procedure operates on all statistic definitions defined in the TDSTATS database.

Teradata recommends running ResyncStats when the following occurs:

- Statistics definitions are automated and dropped as a result of a SQL DROP STATISTICS statement.
- The RunCollectReport external stored procedure reports errors from collections issued on statistics that no longer exist in the dictionary

**Synchronizing the TDSTATS Database**

To synchronize the TDSTATS database, you can call one of the following external stored procedures:

- ResyncStats with *DatabaseName* and *TableName* specified as NULL.
- AutomateStats with *DeleteOrphans* specified as Y.

**Note:**

To copy additional statistics definitions from DBC to the TDSTATS database, use the AutomateStats external stored procedure. For details, see [AutomateStats](#).

**Example: Using ResyncStats**

The following example shows how to synchronize the statistics for the Personnel.Employee table.

```
CALL
TDSTATS.ResyncStats('Personnel','Employee',AutomateId, NumRemovedOrRepaired);
```

**Related Information**

| For more information on the ...                            | See ...                                                                                  |
|------------------------------------------------------------|------------------------------------------------------------------------------------------|
| SQL DROP STATISTICS statement                              | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144. |
| RunCollectReport external stored procedure                 | <a href="#">RunCollectReport</a> .                                                       |
| <i>DeleteOrphans</i> input parameter or copying statistics | <a href="#">AutomateStats</a> .                                                          |

## PreserveAfterRecreate

Marks one or more automated statistics as eligible for having their TDStats stored metadata preserved after their owning table is temporarily dropped and then recreated.

### Syntax

```
REPLACE PROCEDURE TDSTATS.PreserveAfterRecreate (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumPreserved INTEGER
)
...
;
```

### Syntax Elements

#### *DatabaseName*

Database whose automated statistics should be marked for preservation.

#### *ObjectName*

Table within *DatabaseName* whose statistics should be marked for preservation.

#### *ObjectListName*

Object list whose defined databases and tables should be marked for preservation.

#### *NumPreserved*

Number of statistics marked for preservation.

### Usage Notes

The use of this API should be limited to those tables that after being temporarily dropped are subsequently recreated with the exact same name, column definitions, and statistics definitions.

#### Note:

Users are cautioned against using this API in table recreation scenarios that do not meet this criteria as the subsequent automated statistics management operations may fail.

To mark all tables within a given database for preservation, callers should specify a value for input parameter *DatabaseName*. To further limit the preservation to a single table, callers should also specify a value for input parameter *ObjectName*.

To mark the set of databases or tables defined within a previously created object list, callers should specify a value for input parameter `ObjectListName`. For information on how to create an `ObjectListName`, see [CreateObjectList](#).

The values of input parameters `DatabaseName` and `ObjectListName` cannot both be NULL. Callers must specify a non-NULL for one of them.

After a table, whose automated statistics are marked for preservation, is dropped and recreated, users must call `AutomateStats` or `ResyncStats` to complete the preservation process. For more information see [AutomateStats](#) and [ResyncStats](#). An example workflow is shown in [Statistics from Dropped and Recreated Objects](#).

If not marked for preservation, the `TDStats` entries for dropped and recreated tables become orphaned from the Data Dictionary and the associated statistics can no longer be managed through Automated Statistics Management APIs. See [Orphaned Statistics](#) for more information.

### Example: Using PreserveAfterRecreate

The following example shows how to mark the automated statistics `Personnel.Employee` table for preservation.

```
CALL TDSTATS.PreserveAfterRecreate('Personnel','Employee', NULL, NumPreserved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumPreserved
-----
          4
```

## DoNotPreserveAfterRecreate

Marks one or more automated statistics as no longer eligible for having their `TDStats` stored metadata preserved after their owning table is temporarily dropped and then recreated. The qualifying automated statistics are then "unpreserved."

### Syntax

```
REPLACE PROCEDURE TDSTATS.DoNotPreserveAfterRecreate (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumNotPreserved INTEGER
)
...
;
```

## Syntax Elements

### ***DatabaseName***

Database whose automated statistics should no longer be marked for preservation.

### ***ObjectName***

Table within *DatabaseName* whose statistics should no longer be marked for preservation.

### ***ObjectListName***

Object list whose defined databases and tables should no longer be marked for preservation.

### ***NumNotPreserved***

Number of statistics marked no longer eligible for preservation.

## Usage Notes

This API is typically called to reverse the actions taken by a prior call to the related API `PreserveAfterRecreate`. For more information, see [PreserveAfterRecreate](#). Unpreserved is the default state for all automated statistics, so it is not necessary to call `DoNotPreserveAfterRecreate` unless it is to undo a prior call to `PreserveAfterRecreate`.

To unpreserve all tables within a given database for preservation, callers should specify a value for input parameter `DatabaseName`. To further limit the action to a single table, callers should also specify a value for input parameter `ObjectName`.

To unpreserve the set of databases or tables defined within a previously created object list, callers should specify a value for input parameter `ObjectListName`. For information on how to create an `ObjectListName`, see [CreateObjectList](#).

The values of input parameters `DatabaseName` and `ObjectListName` cannot both be NULL. Callers must specify a non-NULL for one of them.

If not marked for preservation, the TDStats entries for dropped and recreated tables become orphaned from the Data Dictionary and the associated statistics can no longer be managed through Automated Statistics Management APIs. See [Statistics from Dropped and Recreated Objects](#) for more information.

### **Example: Using DoNotPreserveAfterRecreate**

The following example shows how to mark the automated statistics for the `Personnel.Employee` table as no longer eligible for preservation.

```
CALL TDSTATS.DoNotPreserveAfterRecreate('Personnel','Employee',
NULL, NumNotPreserved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumNotPreserved
```

## Analyzer Open APIs

Use these external stored procedures for analyzing user objects and logged query plans to identify situations where automated statistics management might be improved.

### AnalyzeStats

Analyzes a set of specified objects to find events where statistics management tasks on them might be improved.

#### Syntax

```
REPLACE PROCEDURE TDSTATS.AnalyzeStats (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN NewerThan TIMESTAMP(6) WITH TIME ZONE,
  IN MarkApproved CHAR(1) CHARACTER SET LATIN,
  OUT AnalysisId BIGINT,
  OUT NumEvents INTEGER
)
...
;
```

#### Syntax Elements

##### ***DatabaseName***

Name of the database. *DatabaseName* limits the operation to objects defined in the database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

##### ***TableName***

Name of the table. *TableName* limits the operation to the table you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

**ObjectListName**

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be analyzed.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

**NewerThan**

Date. *NewerThan* limits the operation to tables created after the date you specify.

**MarkApproved**

Possible values:

- Y or NULL. If you specify Y or NULL, the recommended missing statistics are marked approved and included in subsequent calls to PrepCollect and RunCollect. The default is Y.
- N. If you specify N, the recommended missing statistics are marked unapproved. To approve these statistics, you must call ApproveStat.

**AnalysisId**

Results ID for all events generated by this operation.

**NumEvents**

Number of events recorded by this operation.

**Usage Notes**

If you have permissions to grant the STATISTICS privilege on the qualifying user objects, you should also grant that privilege to the TDSTATS database which allows AnalyzeStats to submit EXPLAIN COLLECT STATISTICS statements (see [Granting Required Privileges](#)).

AnalyzeStats results are made available to subsequent calls to the PrepCollect external stored procedure, which incorporates event information when generating collection commands.

To display the AnalyzeStats results, you can call AnalyzeStatsReport.

If logged query data from the database query log (DBQL) is unavailable, you can call AnalyzeStatsUsage to incorporate recommendations from the Query Optimizer.

**Analyzing Statistics****Note:**

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To analyze ...                                                                        | You can specify a value for ...                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| statistics for a particular database                                                  | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                                        |
| statistics for a particular table                                                     | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                              |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <p><i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |
| only newly created tables                                                             | <i>NewerThan</i> .                                                                                                                                                                                                                                                                                                                                                                           |

## Wildcard Characters

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

## Example: Using AnalyzeStats

The following example shows how to analyze the statistics on the Personnel database for objects created after January 1, 2010.

```
CALL AnalyzeStats ('Personnel', NULL,
NULL, '2010-10-01 00:00:00.00', 'N', AnalysisId, NumEvents);
*** Procedure has been executed.
*** Total elapsed time was 10 seconds.
AnalysisId  NumEvents
-----
1           2
```

## Related Information

| For more information on ...                                                                         | See ...                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| the STATISTICS privilege                                                                            | <a href="#">Granting Required Privileges.</a>                                                                                                                                                                                        |
| the EXPLAIN COLLECT STATISTICS request                                                              | <ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Request and Transaction Processing</i>, B035-1142.</li> </ul> |
| how PrepCollect incorporates event information when generating collection commands                  | <a href="#">PrepCollect.</a>                                                                                                                                                                                                         |
| the AnalyzeStats results                                                                            | <a href="#">AnalyzeStatsReport.</a>                                                                                                                                                                                                  |
| the Analyzer-related external stored procedure that is recommended when query log data is available | <a href="#">AnalyzeStatsUsage.</a>                                                                                                                                                                                                   |

## AnalyzeStatsUsage

Analyzes objects referenced within logged query plans to find conditions where automated statistics management tasks on them might be improved.

### Syntax

```

REPLACE PROCEDURE TDSTATS.AnalyzeStatsUsage (
  IN LogStartTime TIMESTAMP(6) WITH TIME ZONE,
  IN LogEndTime TIMESTAMP(6) WITH TIME ZONE,
  IN LogDatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN ApplName VARCHAR(128) CHARACTER SET UNICODE,
  IN UserName VARCHAR(128) CHARACTER SET UNICODE,
  IN AcctString VARCHAR(128) CHARACTER SET UNICODE,
  IN QBName VARCHAR(128) CHARACTER SET UNICODE,
  IN QBValue VARCHAR(256) CHARACTER SET UNICODE,
  IN QueryListName VARCHAR(128) CHARACTER SET UNICODE,
  IN MarkApproved CHAR(1) CHARACTER SET LATIN,
  OUT AnalysisId BIGINT,
  OUT NumEvents INTEGER
)
...
;

```



## Syntax Elements

### ***LogStartTime***

Start time. *LogStartTime* limits the analysis to queries logged after the time you specify.

### ***LogEndTime***

End time. *LogEndTime* limits the analysis to queries logged before the time you specify.

### ***LogDatabaseName***

Name of the database containing the log tables.

If you specify the *LogDatabaseName* value as PDCRDATA, the operation automatically locates the valid query log tables.

If you specify the *LogDatabaseName* value as a database other than DBC or PDCRDATA, the operation assumes the log tables of the database specified are identical in name and structure to those in the DBC database, namely the DBQLogTbl and DBQLXMLTbl tables.

If you specify *LogDatabaseName* as NULL, DBC is the default database.

### ***DatabaseName***

User database. *DatabaseName* limits the analysis to referenced objects defined in the user database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***TableName***

Referenced user table. *TableName* limits the analysis to the referenced user table you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***ObjectListName***

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be analyzed.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***AppName***

Application name. *AppName* limits the analysis to logged queries with the name of the application you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***UserName***

User name. *UserName* limits the operation to logged queries with the user name you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***AcctString***

Account string. *AcctString* limits the operation to logged queries with the account string you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***QBName***

### ***QBValue***

Query band pair consisting of the following:

*QBName*=*QBValue*

*QBName* limits the operation to logged queries you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***QueryListName***

Name of the query list. *QueryListName* limits the operation to the queries defined in a list previously created by the AddQueryList external stored procedure.

When you specify a value for *QueryListName*, the following input parameters must be NULL:

- *AppName*
- *UserName*
- *AcctString*
- *QBName*
- *QBValue*

For more information, see [AddQueryList](#).

### ***MarkApproved***

Possible values:

- Y or NULL. If you specify Y or NULL, the recommended missing statistics are marked approved and included in subsequent calls to PrepCollect and RunCollect. The default is Y.
- N. If you specify N, the recommended missing statistics are marked unapproved. To approve these statistics, you must call ApproveStat.

**AnalysisId**

Results ID for all events generated by this operation.

**NumEvents**

Number of events recorded by this operation.

**Usage Notes**

If you have permissions to grant the STATISTICS privilege on the qualifying user objects, you should also grant that privilege to the TDSTATS database which allows AnalyzeStatsUsage to submit EXPLAIN COLLECT STATISTICS statements (see [Granting Required Privileges](#)).

Before using AnalyzeStatsUsage, you must enable DBQL while executing queries during the specified time period.

By default, the Analyzer operation assumes the DBQL tables reside within DBC, but you can specify an alternate database that contains archived query logs.

The AnalyzeStatsUsage results are written to the AnalyzerHistoryTbl table in the form of events, such as recommended missing statistics and identified stale statistics. The AnalyzeStatsUsage results are made available to subsequent calls to PrepCollect.

To display the AnalyzeStatsUsage results, you can call AnalyzeStatsUsageReport.

**When to Use AnalyzeStatsUsage**

AnalyzeStatsUsage incorporates logged recommendations from the Query Optimizer. It also performs a workload-level analysis to identify the subset of the Query Optimizer recommendations that occur frequently across the query log and in situations where they may be of most benefit (for example, query steps with inaccurate spool size estimates).

Teradata recommends that you use the AnalyzeStatsUsage external stored procedure when query log data is available and AnalyzeStats when it is not. For more information, see [AnalyzeStats](#).

**Enabling the STATSUSAGE and XMLPLAN Logging Options**

For a more complete analysis, Teradata recommends that you enable both the BEGIN QUERY LOGGING XMLPLAN and STATSUSAGE options. However, query logs with only STATSUSAGE data is sufficient.

You must enable the STATSUSAGE and XMLPLAN options for:

- Recently automated queries referencing databases or tables.
- A significantly changed query workload.

## Analyzing Statistics

### Note:

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To analyze statistics for a ...                                                     | You can specify a value for ...                                                                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| particular database                                                                 | <i>DatabaseName</i>                                                                                                                                                                                                                                                                                                                                                                                        |
| particular table                                                                    | both <i>DatabaseName</i> and <i>TableName</i>                                                                                                                                                                                                                                                                                                                                                              |
| list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <p><i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |

## Wildcard Characters

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

### Example: Using AnalyzeStatsUsage

The following example shows how to analyze the statistics usage in the Personnel database for queries logged over a seven-day period.

```
CALL TDSTATS.AnalyzeStatsUsage ('2010-10-01 00:00:00.000000+00:00','2010-10-08
00:00:00.000000+00:00', NULL,
'Personnel',NULL,NULL,NULL,NULL,NULL,NULL,NULL,'Y',AnalysisId,NumEvents);
*** Procedure has been executed.
*** Total elapsed time was 10 seconds.
```

|            |           |
|------------|-----------|
| AnalysisId | NumEvents |
| -----      | -----     |
| 2          | 4         |

## Related Information

| For more information on ...                    | See ...                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| the STATISTICS privilege                       | <a href="#">GrantPrivileges.</a>                                                                                                                                                                                                                                                                               |
| the EXPLAIN COLLECT STATISTICS request         | <ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - SQL Request and Transaction Processing</i>, B035-1142.</li> </ul>                                                                           |
| displaying the AnalyzeStatsUsage results       | <a href="#">AnalyzeStatsUsageReport.</a>                                                                                                                                                                                                                                                                       |
| the DBC.DBQLogTbl table                        | <i>Teradata Vantage™ - Database Administration</i> , B035-1093.                                                                                                                                                                                                                                                |
| the STATSUSAGE and XMLPLAN logging options     | <ul style="list-style-type: none"> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - Database Administration</i>, B035-1093.</li> <li>• <i>Teradata Vantage™ - SQL Request and Transaction Processing</i>, B035-1142.</li> </ul> |
| the Analyzer-related external stored procedure | <a href="#">AnalyzeStats.</a>                                                                                                                                                                                                                                                                                  |

## CleanupStats

Identifies statistics not used by query optimization for a specified period of time according to the system maintained activity counters and optionally reactivates them.

### Syntax

```

REPLACE PROCEDURE TDSTATS.CleanupStats (
  IN OlderThan TIMESTAMP(6) WITH TIME ZONE,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN MarkInactive CHAR(1) CHARACTER SET LATIN,
  OUT AnalysisId BIG,
  OUT NumEvents INTEGER
)
...
;

```

## Syntax Elements

### ***OlderThan***

Time. *OlderThan* identifies statistics that have not been used since the time you specify as inactive.

### ***DatabaseName***

Name of the database. *DatabaseName* limits the cleanup operation to statistics within the database you specify.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***TableName***

Name of the table within the specified *DatabaseName*. *TableName* limits the cleanup operation to the statistics defined on the table you specify within *DatabaseName*.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

### ***ObjectListName***

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be cleaned.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***MarkInactive***

Possible values:

- Y. If you specify Y, statistics are automatically marked and not considered in subsequent calls to the PrepCollect external stored procedure.  
Any statistics identified for reactivation are automatically reactivated for PrepCollect. For more information, see [PrepCollect](#).
- N. If you specify N, statistics remain in the same active or inactive state. You must call InactivateStat to inactivate the statistics or ActivateStat to reactivate them. For more information on these external stored procedures, see [InActivateStat](#) and [ActivateStat](#). This is the default.

### ***AnalysisId***

Results ID for all events generated by this operation.

**NumEvents**

Number of statistics inactivated or reactivated as a result of the cleaning.

**Usage Notes**

The CleanupStats results are written to the AnalyzerHistoryTbl table. To display the results, you can call CleanupStatsReport.

**Enabling Object Use Counts for Statistics**

Before calling CleanupStats, you must enable object use counts for statistics for queries accessing the specified *DatabaseName* and *TableName*.

**Note:**

The USECOUNT logging option in the BEGIN QUERY LOGGING statement must be enabled long enough to evaluate the value specified for the *OlderThan* input parameter.

**Identifying Statistics**

**Note:**

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To identify only the statistics not used for query optimization for a ...             | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| particular database                                                                   | <i>DatabaseName</i>                                                                                                                                                                                                                                                                                                                                                               |
| particular table                                                                      | both <i>DatabaseName</i> and <i>TableName</i>                                                                                                                                                                                                                                                                                                                                     |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.<br><br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

**Wildcard Characters**

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

### Example: Using CleanupStats

The following example shows how to identify the statistics not used on the Personnel.Employee database in the last year.

```
CALL TDSTATS.CleanupStats(,(CAST(CURRENT_DATE - INTERVAL '1' YEAR AS
TIMESTAMP(6)), 'Personnel', 'Employee', NULL, 'N', AnalysisId, NumEvents);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
AnalysisId    NumEvents
-----
          6          3
```

### Related Information

| For more information on ...                                           | See ...                                                                                  |
|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| displaying the CleanupStats results report                            | <a href="#">CleanupStatsReport</a> .                                                     |
| the SQL BEGIN QUERY LOGGING statement and the USECOUNT logging option | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144. |

## AnalyzeStatsReport

Displays a summary of events recorded from a prior invocation of the AnalyzeStats external stored procedure.

### Syntax

```
REPLACE PROCEDURE TDSTATS.AnalyzeStatsReport (
  IN AnalysisId BIGINT
)
...
;
```



## Syntax Elements

### AnalysisId

Result ID from a prior call to the AnalyzeStats external stored procedure.

## Output

| Column               | Description                                                                                                                                                                                                                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EventType            | Possible values: <ul style="list-style-type: none"> <li>• M means a missing statistic whose initial collection is recommended.</li> <li>• S means a stale statistic whose recollection is recommended.</li> </ul>                                                                                   |
| EventDescription     | Text description of the event.                                                                                                                                                                                                                                                                      |
| EventTimeStamp       | Time in which the event occurred.                                                                                                                                                                                                                                                                   |
| DatabaseName         | Database on which the analysis event was recorded.                                                                                                                                                                                                                                                  |
| TableName            | Table on which the analysis event was recorded.                                                                                                                                                                                                                                                     |
| IndexNumber          | Index number on which the event was recorded.<br><b>Note:</b><br>For non-index objects, the value is NULL.                                                                                                                                                                                          |
| IndexName            | User assigned name for the index, if any.                                                                                                                                                                                                                                                           |
| FieldNames           | Column or expression names on which the event was recorded.<br><b>Note:</b><br>Multiple names are comma separated.                                                                                                                                                                                  |
| MissingCritical      | Possible values: <ul style="list-style-type: none"> <li>• Y means the missing statistic is deemed critical and stored in the TDSTATS.StatsTbl table.</li> <li>• N means the missing statistic is not deemed critical and has only been recorded in the TDSTATS.AnalyzerHistoryTbl table.</li> </ul> |
| MissingRank          | Relative rank among all missing events.                                                                                                                                                                                                                                                             |
| Approved             | Flag indicates the corresponding collection is approved for subsequent calls to PrepCollect.<br><b>Note:</b><br>The Approved value is valid only to those statistics where the MissingCritical value is Y. For more information, see the MissingCritical column.                                    |
| Stale                | Possible values: <ul style="list-style-type: none"> <li>• Y means statistic is stale and needs to be refreshed.</li> <li>• N means statistic is not stale or cannot be determined.</li> </ul>                                                                                                       |
| EstPercentDataChange | Estimated percent of data changed since the last collection.                                                                                                                                                                                                                                        |

| Column          | Description                                                                                                                                                                                                                                                |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | <b>Note:</b><br>The EstPercentDataChange output parameter value is NULL when the EventType value is M or when the EstPercentDataChange output parameter value is not a reliable estimate.                                                                  |
| Age             | Number of days since the statistic was last collected.                                                                                                                                                                                                     |
| Threshold       | Dictionary defined THRESHOLD setting for the statistic encoded as a text signature.<br>This value may be defined with the Age and EstPercentDataChange column settings. For details, see the EstPercentDataChange and Age columns.                         |
| AvgImportance   | System assigned average importance of the statistic.                                                                                                                                                                                                       |
| NeedsAutomation | Possible values: <ul style="list-style-type: none"> <li>• Y means the analyzed existing statistic is not automated.</li> <li>• N means analyzed existing statistic is automated.</li> </ul> To automate the statistic, see <a href="#">AutomateStats</a> . |
| StatsId         | Dictionary assigned ID for the analyzed statistic as recorded in the StatsID column of the DBC.StatsTbl table.                                                                                                                                             |
| StatsName       | User assigned name for the statistic, if any.                                                                                                                                                                                                              |
| SCOID           | Automated Statistic Management assigned ID for the statistic as recorded in the TDSTATS.StatsTbl.SCOID column.                                                                                                                                             |
| AnalysisId      | Automated Statistic Management assigned ID for the history results from this operation.                                                                                                                                                                    |

## Usage Notes

This external stored procedure provides a description of each event along with the name or ID of the objects on which the corresponding statistics are defined or found missing.

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Example: Using AnalyzeStatsReport

The following example reports the summary of events recorded on the Personnel database from a prior invocation of AnalyzeStats.

```
CALL TDSTATS.AnalyzeStatsReport(1);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
*** ResultSet# 1 : 2 rows returned by "TDSTATS.ANALYZESTATSREPORT".
EventType EventDescription      DataBaseName  ObjectName  IdxNum  FieldNames
-----
M          MISSING STATS ON INDEX  PERSONNEL    CHARGES     1      EMPNO, PROJ_ID
M          MISSING STATS ON INDEX  PERSONNEL    CHARGES     4      PROJ_ID
```

## AnalyzeStatsUsageReport

Displays a summary of events recorded from a prior call to the `AnalyzeStatsUsage` external stored procedure (see [AnalyzeStatsUsage](#)).

### Syntax

```
REPLACE PROCEDURE TDSTATS.AnalyzeStatsUsageReport (
  IN AnalysisId BIGINT
)
...
;
```

### Syntax Elements

#### *AnalysisId*

Result ID from a prior call to the `AnalyzeStatsUsage` external stored procedure.

### Output

| Column           | Description                                                                                                                                                                                                                                                                         |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EventType        | Possible values: <ul style="list-style-type: none"> <li>• M means missing statistic whose initial collection is recommended.</li> <li>• S means stale existing statistic whose recollection is recommended.</li> <li>• U means used statistic during query optimization.</li> </ul> |
| EventDescription | Text description of the event or action.                                                                                                                                                                                                                                            |
| EventFreq        | Frequency of the event across the query log.                                                                                                                                                                                                                                        |
| EventTimeStamp   | Time in which the event occurred.                                                                                                                                                                                                                                                   |
| DatabaseName     | Database on which the analysis event was recorded.                                                                                                                                                                                                                                  |
| TableName        | Table on which the analysis event was recorded                                                                                                                                                                                                                                      |

| Column               | Description                                                                                                                                                                                                                                                                                                     |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IndexNumber          | Index number on which the event was recorded.<br><b>Note:</b><br>The value is NULL for non-index objects.                                                                                                                                                                                                       |
| IndexName            | User assigned name for the index, if any.                                                                                                                                                                                                                                                                       |
| FieldNames           | Column or expressions on which the event was recorded.<br><b>Note:</b><br>Multiple names are comma separated.                                                                                                                                                                                                   |
| MissingCritical      | Possible values:<br><ul style="list-style-type: none"> <li>• Y means missing statistics were deemed critical and have been stored in the TDSTATS.StatsTbl table.</li> <li>• N means missing statistics were not deemed critical and have only been recorded in the TDSTATS.AnalyzerHistoryTbl table.</li> </ul> |
| MissingRank          | Relative rank among all missing events.                                                                                                                                                                                                                                                                         |
| Approved             | Flag indicates the corresponding collection is approved for subsequent calls to the PrepCollect external stored procedure.<br><b>Note:</b><br>This value is valid only for those statistics where the MissingCritical value is Y. For more information, see the MissingCritical column.                         |
| Stale                | Possible values:<br><ul style="list-style-type: none"> <li>• Y means statistic is stale and needs to be refreshed.</li> <li>• N means statistic is not stale or cannot be determined.</li> </ul>                                                                                                                |
| EstPercentDataChange | Estimated percent of data changed since the last collection.<br><b>Note:</b><br>EstPercentDataChange returns NULL when the EventType value is M or when the EstPercentDataChange value is not a reliable estimate.                                                                                              |
| Age                  | Number of days since the statistic was last collected.                                                                                                                                                                                                                                                          |
| Threshold            | Dictionary defined THRESHOLD setting for the statistic encoded as a text signature.<br>The Threshold column may be defined with the Age and EstPercentDataChange column settings. For details, see the EstPercentDataChange and Age columns.                                                                    |
| CardEstErrFreq       | Number of occurrences of cardinality estimation error for this observed event.                                                                                                                                                                                                                                  |
| AvgImportance        | Average recorded importance across query log.                                                                                                                                                                                                                                                                   |
| MinImportance        | Minimum recorded importance across query log.                                                                                                                                                                                                                                                                   |
| MaxImportance        | Maximum recorded importance across query log.                                                                                                                                                                                                                                                                   |

| Column            | Description                                                                                                                                                                                                                                                            |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MarkedInvalid     | Flag indicates that the statistic is no longer usable as determined by the query optimizer.                                                                                                                                                                            |
| MarkedSummaryOnly | Flag indicates that only summary statistics were needed during query optimization. For more information about the SQL COLLECT STATISTICS SUMMARY option, see <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.                  |
| NeedsAutomation   | Possible values: <ul style="list-style-type: none"> <li>• Y means that the analyzed existing statistic is not automated</li> <li>• N means that the analyzed existing statistic is automated</li> </ul> To automate the statistic, see <a href="#">AutomateStats</a> . |
| StatsId           | Dictionary assigned ID for the statistic as recorded in the DBC.StatsTbl. StatsId column.                                                                                                                                                                              |
| StatsName         | User assigned name for the statistic, if any.                                                                                                                                                                                                                          |
| SCOID             | Automated Statistic Management assigned ID for the statistic as recorded in the SCOID column of the TDSTATS.StatsTbl table.                                                                                                                                            |
| AnalysisId        | Automated Statistic Management assigned ID for the history results from this operation.                                                                                                                                                                                |

## Usage Notes

For each distinct event, the AnalyzeStatsUsage external stored procedure reports the total number of occurrences across the specified logged period along with the name or ID of the object on which the event applies.

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Example: Using AnalyzeStatsUsageReport

The following example shows how to report a summary of events recorded on the Personnel database from a prior invocation of AnalyzeStatsUsage.

```
CALL TDSTATS.AnalyzeStatsUsageReport(2);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
*** ResultSet# 1 : 3 rows returned by "TDSTATS.ANALYZESTATSUSAGEREPORT".
```

| EventType   | EventDescription       | DataBaseName | TableName         |
|-------------|------------------------|--------------|-------------------|
| IndexNumber | FieldNames             |              |                   |
| -----       | -----                  | -----        | -----             |
| M           | MISSING STATS ON       |              |                   |
| INDEX       | PERSONNEL CHARGES      | 1            | EMPNO, PROJ_ID    |
| M           | MISSING STATS ON INDEX | PERSONNEL    | CHARGES 4 PROJ_ID |
| S           | STALENESS DETECTED     | PERSONNEL    | EMPOYEE ? NAME    |

## CleanupStatsReport

Displays a summary of events recorded from a prior call to the CleanupStats external stored procedure.

### Syntax

```
REPLACE PROCEDURE TDSTATS.CleanupStatsReport (
  IN AnalysisId BIGINT
)
...
;
```

### Syntax Elements

#### *AnalysisId*

Result ID from a prior call to the CleanupStats external stored procedure. For more information, see [CleanupStats](#).

### Output

| Column           | Description                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EventType        | Event type: <ul style="list-style-type: none"> <li>• C means the event is identified for cleaning.</li> <li>• R means the event is identified for reactivation.</li> </ul> |
| EventDescription | Text description of the event or action.                                                                                                                                   |
| EventTimeStamp   | Time at which the event occurred.                                                                                                                                          |
| DatabaseName     | Database on which the cleanup event was recorded.                                                                                                                          |
| ObjectName       | Table on which the cleanup event was recorded.                                                                                                                             |
| IndexNumber      | Index number on which the cleanup event was recorded. <p><b>Note:</b><br/>This value is NULL for non-index objects.</p>                                                    |

| Column            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| FieldNames        | Column or expression name on which the event was recorded.<br><b>Note:</b><br>Multiple names are separated by a comma.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| MarkedInactive    | If EventType is C, the possible values are: <ul style="list-style-type: none"> <li>• Y means the identified statistic is marked ineligible for future PrepCollect calls.</li> <li>• N means the identified statistic is marked eligible for future PrepCollect calls.</li> </ul> If EventType is R, the possible values are: <ul style="list-style-type: none"> <li>• Y means the identified statistic is marked eligible for future PrepCollect calls.</li> <li>• N means the identified statistic is marked ineligible for future PrepCollect calls.</li> </ul> For more information, see <a href="#">PrepCollect</a> . |
| LastAccessedByOpt | Last recorded usage by the query optimizer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| StatsId           | Dictionary assigned ID for the statistic as recorded in the DBC.StatsTbl.StatsId column.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| StatsName         | User assigned name for the statistic, if any.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| SCOID             | Automated Statistics Management assigned ID for the statistic as recorded in the TDSTATS.StatsTbl.SCOID column.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| AnalysisId        | Automated Statistics Management assigned ID for the history results from this CleanupStats operation.<br>For more information about this external stored procedure, see <a href="#">CleanupStats</a> .                                                                                                                                                                                                                                                                                                                                                                                                                    |

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Example: Using CleanupStatsReport

The following example shows the summary of events for the Personnel database from a prior invocation of CleanupStats.

```
CALL TDSTATS.CleanupStatsReport(2);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
*** ResultSet# 1 : 3 rows returned by "TDSTATS.CLEANUPSTATSUSAGEREPORT".
EventType  EventDescription                      DataBaseName  TableName  IndexNumber
FieldNames
```

|       |                                 |           |         |   |      |
|-------|---------------------------------|-----------|---------|---|------|
| ----- |                                 |           |         |   |      |
| ----- |                                 |           |         |   |      |
| C     | STATS IDENTIFIED FOR CLEANING   | PERSONNEL | CHARGES |   |      |
| 1     | EMPNO, PROJ_ID                  |           |         |   |      |
| C     | STATS IDENTIFIED FOR CLEANING   | PERSONNEL | CHARGES |   |      |
| 4     | PROJ_ID                         |           |         |   |      |
| R     | STATS IDENTIFIED FOR ACTIVATION | PERSONNEL | EMPOYEE | ? | NAME |

## ApproveStat

Marks an Analyzer missing statistic recommendation as being user approved making it eligible for future automated collections.

### Syntax

```
REPLACE PROCEDURE TDSTATS.ApproveStat (  
  IN SCOID BIGINT,  
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,  
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,  
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,  
  OUT NumApproved INTEGER  
)  
  ...  
;
```

### Syntax Elements

#### SCOID

TDSTATS ID for a missing statistics recommendation from an Analyze-related operation or an existing statistic copied from an Automate-related operation.

#### DatabaseName

Name of the database. *DatabaseName* approves all statistics defined on the database you specify.

#### TableName

Name of the table within the specified *DatabaseName*. *TableName* limits the approval to the statistics defined on the table you specify within *DatabaseName*.

#### ObjectListName

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be approved.



For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***NumApproved***

Number of approved statistics that were previously unapproved.

## **Usage Notes**

Depending on how Analyze-related external stored procedures are called, any:

- Resulting recommendations for new statistics may remain unapproved until ApproveStat is performed on them.
- Existing statistics copied to the TDSTATS database may also remain unapproved until ApproveStat is performed on them.

To determine which recommended missing statistics are still marked unapproved after an Analyzer-related operation, you can call the following external stored procedures:

- AnalyzeStatsReport
- AnalyzeStatsUsageReport
- SelectAutomatedStats and examine the ApprovedStat result column.

To determine which imported statistics are still marked unapproved after an Automate-related operation, you can call the following external stored procedures:

- AutomateReport
- SelectAutomatedStats

## **Approving Statistics Recommendations**

### **Note:**

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To approve ...                                           | You must specify a value for ...                                                                                                                              |
|----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single statistic recommendation                        | <i>SCOID</i> .<br><b>Note:</b><br>When specifying a value for <i>SCOID</i> , you do not need to specify a value for <i>DatabaseName</i> or <i>TableName</i> . |
| all statistics recommendations for a particular database | <i>DatabaseName</i> .                                                                                                                                         |
| all statistics recommendations for a particular table    | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                               |

| To approve ...                                                                        | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.<br><br><b>Note:</b><br><i>ObjectName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

**Example: Using ApproveStat**

The following example shows how to approve all unapproved statistic recommendations on the Employee table.

```
CALL TDSTATS.ApproveStat(NULL,'Personnel','Employee',NULL, NumApproved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumApproved
-----
1
```

**DisapproveStat**

Marks an analyzer missing statistic recommendation as being user disapproved making it ineligible for future automated collections and prevents it from being generated by future Analyzer-related calls.

**Syntax**

```
REPLACE PROCEDURE TDSTATS.DisapproveStat (
  IN SCOID BIGINT,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumDisapproved INTEGER
)
...
;
```

## Syntax Elements

### **SCOID**

TDSTATS ID for a missing statistics recommendation.

### **DatabaseName**

Name of the database. *DatabaseName* approves all statistics defined on the database you specify.

### **TableName**

Name of the table within the specified *DatabaseName*. *TableName* limits the approval to the statistics defined on the table you specify within *DatabaseName*.

### **ObjectListName**

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be approved.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### **NumDisapproved**

Number of statistics marked disapproved.

## Usage Notes

This operation applies only to missing statistic recommendations with a TDSTATS.StatsTbl.State column value of M. You can use Teradata Studio or Teradata Studio Express to view details of the TDSTATS.StatsTbl.State column.

You can call the DeAutomateStats external stored procedure to remove an already collected statistic from the TDSTATS database or inactivate it by calling the InActivateStat external stored procedure.

You can call the following external stored procedures to determine which newly recommended statistics are still unapproved from the Analyzer-related operations:

- AnalyzeStatsReport or AnalyzeStatsUsageReport
- SelectAutomatedStats and examine the ApprovedStat column

## Disapproving Statistics Recommendations

### **Note:**

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To disapprove ...                                                                     | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single statistics recommendation                                                    | <i>SCOID</i> .<br><b>Note:</b><br>When specifying a value for <i>SCOID</i> , you do not need to specify a value for the <i>DatabaseName</i> or <i>TableName</i> .                                                                                                                                                                                                                           |
| all statistics recommendations for a specified database                               | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                                       |
| all statistics for a specified table                                                  | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                             |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.<br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

### Example: Using DisapproveStat

The following example shows how to disapprove an individual statistic recommendation on the Personnel.Employee table whose TDSTATS database ID is 101.

```
CALL TDSTATS.DisapproveStat(101,'Personnel','Employee', NULL, NumDisapproved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumDisapproved
-----
1
```

## InActivateStat

Marks an automated statistic inactive because of no observed query usage and disables it from future collections by the PrepCollect and RunCollect external stored procedures.

### Syntax

```
REPLACE PROCEDURE TDSTATS.InActivateStat (
  IN SCOID BIGINT,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
```

```
    OUT NumInActivated INTEGER
  )
  ...
;
```

Syntax Elements

**SCOID**  
TDSTATS ID for an individual statistic.

**DatabaseName**  
Name of the database. *DatabaseName* approves all statistics defined on the database you specify.

**TableName**  
Name of the table within the specified *DatabaseName*. *TableName* limits the approval to the statistics defined on the table you specify within *DatabaseName*.

**ObjectListName**  
Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be approved.  
For more information about this input parameter, see the usage notes or [CreateObjectList](#).

**NumInActivated**  
Number of statistics marked inactive that were previously active.

Usage Notes

Depending on how the CleanupStats external stored procedure is called, any identified inactive statistics may remain active until the InActivateStat external stored procedure is called on them. For more information about this external stored procedure, see [CleanupStatsReport](#).

Inactivating Statistics

**Note:**  
If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To inactivate ...  | You must specify a value for ... |
|--------------------|----------------------------------|
| a single statistic | SCOID.                           |

| To deactivate ...                                                                     | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                       | <b>Note:</b><br>When specifying a value for SCOID, you do not need to specify a value for the <i>DatabaseName</i> or <i>TableName</i> .                                                                                                                                                                                                                                           |
| all statistics for a particular database                                              | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                             |
| all statistics for a particular table                                                 | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                   |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.<br><br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

### Example: Using InActivateStat

The following example shows how to deactivate an automated statistic on the Personnel.Employee table whose TDSTATS database ID is 101.

```
CALL TDSTATS.InActivateStat(101,'Personnel','Employee', NULL, NumInActivated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumInActivated
-----
1
```

## ActivateStat

Reactivates a previously inactivated statistic to reflect observed query usage and makes it eligible again for future collections via the PrepCollect and RunCollect external stored procedures.

### Syntax

```
REPLACE PROCEDURE TDSTATS.ActivateStat (
  IN SCOID BIGINT,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN MarkPermanent CHAR(1) CHARACTER SET LATIN,
  OUT NumActivated INTEGER
```

```
)
...
;
```

## Syntax Elements

### **SCOID**

TDSTATS ID for an individual statistic.

### **DatabaseName**

Name of the database. *DatabaseName* approves all statistics defined on the database you specify.

### **TableName**

Name of the table within the specified *DatabaseName*. *TableName* limits the approval to the statistics defined on the table you specify within *DatabaseName*.

### **ObjectListName**

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be approved.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### **MarkPermanent**

Possible values:

- Y. If you specify Y, the statistic is permanently marked active such that future calls to the CleanupStats external stored procedure are not recommended for inactivation.
- N or NULL. If you specify N or NULL, the statistic remains a candidate for inactivation by future calls to the CleanupStats external stored procedure. The default value is N.

### **NumActivated**

Number of statistics marked active that were previously inactive or not permanently active.

## Usage Notes

You can use the ActivateStat external stored procedure to mark a still active or inactive statistic as permanently active so that the statistic is removed from future consideration by CleanupStats. For more information about this external stored procedure, see [CleanupStats](#).

## Reactivating Statistics

### Note:

If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To reactivate ...                                                                     | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single statistic                                                                    | <i>SCOID</i> .<br><b>Note:</b><br>When specifying a value for <i>SCOID</i> , you do not need to specify a value for the <i>DatabaseName</i> or <i>TableName</i> .                                                                                                                                                                                                             |
| all statistics for a particular database                                              | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                         |
| all statistics for a particular table                                                 | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                               |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.<br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

### Example: Using ActivateStat

The following example shows how to reactivate a previously inactivated statistic on the Personnel.Employee table whose TDSTATS database ID is 101.

```
CALL TDSTATS.ActivateStat(101,'Personnel','Employee', NULL, NULL, NumActivated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumActivated
-----
1
```

## Collect Open APIs

Use these external stored procedures for preparing and submitting the SQL COLLECT STATISTICS statements on the automated statistic definitions stored in the TDSTATS database. For more information about the SQL COLLECT STATISTICS statement, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.



## PrepCollect

Prepares a prioritized list of SQL COLLECT STATISTICS statements based on the automated statistics definitions stored in the TDSTATS database.

### Syntax

```
REPLACE PROCEDURE TDSTATS.PrepCollect (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  [ IN CmdListName VARCHAR(128) CHARACTER SET UNICODE, ]
  OUT NumStmtsPrepped INTEGER,
  OUT NumStatsPrepped INTEGER,
  OUT CmdListID BIGINT
)
...
;
```

### Syntax Elements

#### *DatabaseName*

Name of the database. *DatabaseName* allows you to specify the database whose statistics should be prepared for collection.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

#### *TableName*

Name of a table. *TableName* allows you to specify the table whose statistics should be prepared for collection.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

#### *ObjectListName*

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose statistics should be prepared for execution.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

**CmdListName**

[Optional] User assigned name for the resulting prepared list of commands.

*CmdListName* can be referred to by its name rather than its system assigned numeric *CmdListID*. If PrepCollect is called again with a specified *CmdListName* that already exists, the list will be overwritten and the original *CmdListId* retained.

You can execute a command list as many times as needed, although Teradata recommends that you call PrepCollect before each call to RunCollect. For more information, see [RunCollect](#).

**NumStmtsPrepped**

Number of COLLECT STATISTICS statements prepared in the list. Each statement includes one or more individual statistics on a table.

**NumStatsPrepped**

Number of individual statistics prepared for collection in the list.

**CmdListID**

System-assigned ID for the prepared list of statements.

**Usage Notes**

By default, PrepCollect batches and issues together all individual statistics defined on a specified table within a single SQL COLLECT STATISTICS statement.

Any objects excluded by a prior call to AddExcludedObject will not have their statistics prepared for collection.

Only statistics marked approved by the user are included in the prepared collection list.

Any statistics marked inactive by a prior call to CleanupStats or InActivateStat will not be prepared for collection.

**Wildcard Characters**

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

Preparing Statistics for Collection

**Note:**  
If you do not specify a value for the following input parameters, the operation is performed on all user created databases in the system.

| To prepare ...                                                                                       | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| all statistics for a particular database for collection                                              | <i>DatabaseName</i> .                                                                                                                                                                                                                                                                                                                                                             |
| all statistics for a particular table for collection                                                 | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                   |
| a list of objects that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> for collection | <i>ObjectListName</i> that was previously created by the CreateObjectList and AddObjectListEntry external stored procedures.<br><br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

Example: Using PrepCollect to Prepare a Named List of Collections for a Database

The following example shows how to prepare a list of collections for the Personnel database and assign the list a name (for example, MyCmdsList).

```
CALL TDSTATS.PrepCollect ('PERSONNEL', NULL, NULL, 'MyCmdsList',
NumStmtsPrepped, NumStatsPrepped, CmdListID);
*** Procedure has been executed.
*** Total elapsed time was 3 seconds.
```

|                 |                 |           |
|-----------------|-----------------|-----------|
| NumStmtsPrepped | NumStatsPrepped | CmdListID |
| -----           | -----           | -----     |
| 20              | 45              | 1         |

Example: Using PrepCollect to Prepare a List of Collections on an Object List

The following example shows how to prepare a list of collections on the object list named FinCashList.

```
CALL TDSTATS.PrepCollect(NULL , NULL, 'FinCashList', NULL, NumStmtsPrepped,
NumStatsPrepped, CmdListID);
```

## Related Information

| For more information on ...                        | See ...                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| displaying and customizing the prepared collection | <a href="#">RunCollect</a> .<br>You can also refer to the following external stored procedures: <ul style="list-style-type: none"> <li>• <a href="#">SelectPreparedCollects</a>.</li> <li>• <a href="#">AddPreparedCollect</a>.</li> <li>• <a href="#">RemovePreparedCollect</a>.</li> <li>• <a href="#">PrioritizePreparedCollect</a>.</li> </ul> |
| statistics marked approved                         | <a href="#">ApproveStat</a> .                                                                                                                                                                                                                                                                                                                      |
| statistics marked inactive                         | <ul style="list-style-type: none"> <li>• <a href="#">InActivateStat</a>.</li> <li>• <a href="#">CleanupStats</a>.</li> </ul>                                                                                                                                                                                                                       |
| excluded objects                                   | <a href="#">AddExcludedObject</a> .                                                                                                                                                                                                                                                                                                                |

## RunCollect

Executes a prioritized list of SQL COLLECT STATISTICS statements as prepared by a prior call to PrepCollect.

### Syntax

```

REPLACE PROCEDURE TDSTATS.RunCollect (
  IN CmdListID BIGINT,
  IN CmdListName VARCHAR(128) CHARACTER SET UNICODE,
  IN Duration INTEGER,
  IN Resume CHAR(1) CHARACTER SET LATIN,
  OUT NumStmtsSubmitted INTEGER,
  OUT NumStatsSubmitted INTEGER,
  OUT NumStatsCollected INTEGER,
  OUT NumErrors INTEGER,
  OUT DurationExpired CHAR(1) CHARACTER SET LATIN,
  OUT RunID BIGINT
)
...
;

```

### Syntax Elements

#### *CmdListID*

Command list ID assigned from a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

***CmdListName***

User-named commands list as specified in a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

***Duration***

Maximum time limit in minutes for submitting collections within the list.

The valid range is 1 to 4320 (or 72 hours).

If you specify NULL, the duration is unlimited and RunCollect will not complete until all collection commands in the list have finished executing. After the specified *Duration* expires, no new collections will be submitted, but any already in-progress submissions will be allowed to complete before returning control to the caller.

***Resume***

Possible values:

- Y. If you specify Y, execution is resumed where a prior RunCollect operation left off from the most recent execution of the same commands list.
- N or NULL. If you specify N or NULL, execution starts at the beginning of the command list.

For more information about this input parameter, see the usage notes.

***NumStmtsSubmitted***

Number of COLLECT STATISTICS statements that were submitted. Each statement may batch together multiple statistics on a specified table.

***NumStatsSubmitted***

Number of individual statistics submitted for collection independent of any THRESHOLD settings in effect.

***NumStatsCollected***

Number of individual statistics that were actually collected and were not skipped due to any defined THRESHOLD settings.

***NumErrors***

Number of statements that encountered a failure.

**DurationExpired**

Possible values:

- Y means the *Duration* value specified expired before the RunCollect external stored procedure finished executing the entire commands list.
- N means the *Duration* value specified did not expire.

**RunID**

System-assigned ID for the results associated for this execution or run operation.

**Usage Notes**

Teradata recommends that you call PrepCollect before each call to RunCollect to ensure the latest information regarding missing and stale statistics is incorporated into the command list to be executed. You can choose to re-execute an already prepared list by bypassing subsequent calls to PrepCollect.

If you have permissions to grant the STATISTICS privilege on the qualifying user objects, you must also grant that privilege to the TDSTATS database which allows RunCollect to submit COLLECT STATISTICS statements (see [Granting Required Privileges](#)).

By default, PrepCollect batches all individual statistics for a specified table into a single SQL COLLECT STATISTICS statement.

Unless you specify Y for the *Resume* input parameter, RunCollect submits commands starting at the beginning of the list. If a prior call to RunCollect expired, as determined by the DurationExpired output parameter, you may want to resume the execution on that same list by instructing RunCollect to pick up where it left off in the command list.

You can also call:

- AbortCollect to abort one or more pending or in-progress collections under the submission control of RunCollect.
- RunCollectReport to report on the progress of any RunCollect operating on a specified commands list.
- SelectStatsExecutionHistory to display the historical results of one or more past completed RunCollect calls.

**Example: Using RunCollect**

The following example executes the prepared collections for the command list named, MyCmdsList, and specifies a maximum duration of 2 hours.

```
CALL TDSTATS.RunCollect (NULL,'MyCmdsList',120,'N',NumStmtsSubmitted,
NumStatsSubmitted,NumStatsCollected,NumErrors,DurationExpired,RunID);
*** Procedure has been executed.
*** Total elapsed time was 4 seconds.
NumStmtsSubmitted NumStatsSubmitted NumStatsCollected NumErrors DurationExpired RunID
-----
10 20 15 0 'N' 1
```

## Related Information

| For more information on ...               | See ...                                                                                                                                                                                                                                                                      |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| the PrepCollect external stored procedure | <a href="#">PrepCollect</a> .                                                                                                                                                                                                                                                |
| SQL COLLECT STATISTICS statements         | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.                                                                                                                                                                                     |
| THRESHOLD settings                        | <ul style="list-style-type: none"> <li>• <a href="#">UpdateStatThresholdSetting</a>.</li> <li>• <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li>• <i>Teradata Vantage™ - Database Administration</i>, B035-1093.</li> </ul> |

## ReCollectTable

Prepares and executes the automated collections for a specified table.

### Syntax

```

REPLACE PROCEDURE TDSTATS.ReCollectTable (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN Duration INTEGER,
  OUT NumStmtsSubmitted INTEGER,
  OUT NumStatsSubmitted INTEGER,
  OUT NumStatsCollected INTEGER,
  OUT NumErrors INTEGER,
  OUT DurationExpired CHAR(1) CHARACTER SET LATIN,
  OUT RunID BIGINT
)
  ...
;

```

### Syntax Elements

#### ***DatabaseName***

Name of the database to recollect on.

This input parameter cannot be NULL.

#### ***TableName***

Name of the table within the specified *DatabaseName* to recollect on.

This input parameter cannot be NULL.

***Duration***

Maximum time limit in minutes for submitting collections on the table.

The valid range is 1 to 4320 (or 72 hours).

After the specified *Duration* expires, no new collections can be submitted. However, any already in-progress submissions are allowed to complete before returning control to the caller.

NULL indicates an unlimited duration.

***NumStmtsSubmitted***

Number of COLLECT STATISTICS statements submitted.

Each statement may batch together multiple statistics on the specified table.

***NumStatsSubmitted***

Number of individual statistics submitted for collection independent of any THRESHOLD settings in effect.

***NumStatsCollected***

Number of individual statistics actually collected and not skipped due to any defined THRESHOLD settings.

***NumErrors***

Number of statements that encountered a failure.

***DurationExpired***

Possible values:

- Y means the user specified *Duration* expired before the RunCollect external stored procedure finished executing the commands list.
- N means the user specified *Duration* did not expire before the RunCollect external stored procedure finished executing the commands list.

***RunID***

System-assigned ID for the results associated with this execution or run.

You can use *RunID* to call SelectStatsExecutionHistory to display results from a completed ReCollectTable call.



## Usage Notes

You can call `ReCollectTable` after the completion of events that significantly alter data of a specified table, such as bulk loads.

The functionality of `ReCollectTable` is equivalent to `PrepCollect` followed immediately by a `RunCollect` on an individual table.

`ReCollectTable` batches and issues some or all of the statistics for the specified table within a single SQL `COLLECT STATISTICS` statement.

If you have permissions to grant the `STATISTICS` privilege on the qualifying user objects, you must also grant that privilege to the `TDSTATS` database which allows `ReCollectTable` to submit `COLLECT STATISTICS` statements (see [Granting Required Privileges](#)).

Only statistics marked approved by the user are collected.

Any statistics marked inactive by a prior call to `CleanupStats` or `InActivateStat` are not collected. For more information on these external stored procedures, see [CleanupStats](#) or [InActivateStat](#).

### Example: Using ReCollectTable

```
CALL TDSTATS.ReCollectTable ('PERSONNEL', 'EMPLOYEE', NULL, NumStmtsSubmitted,
NumStatsSubmitted,NumStatsCollected,NumErrors,DurationExpired, RunID);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
```

| NumStmtsSubmitted | NumStatsSubmitted | NumStatsCollected | NumErrors | DurationExpired | RunID |
|-------------------|-------------------|-------------------|-----------|-----------------|-------|
| 1                 | 4                 | 4                 | 0         | N               | 1     |

## RunCollectReport

Displays all pending, in-progress, and completed collections for the most recent execution run operation of a given commands list or the results for a specified completed execution run operation as identified by its *RunID* output parameter.

### Syntax

```
REPLACE PROCEDURE TDSTATS.RunCollectReport (
  IN CmdListID BIGINT,
  IN CmdListName VARCHAR(128) CHARACTER SET UNICODE,
  IN RunID BIGINT
)
```

```
...  
;
```

## Syntax Elements

### ***CmdListID***

System-assigned commands list ID.

For more information about this input parameter, see the usage notes.

### ***CmdListName***

User-assigned commands list name.

For more information about this input parameter, see the usage notes.

### ***RunID***

System-assigned ID for the results from a given execution of RunCollect.

To show the results from a specific execution run, you must specify a value for *RunID*.

If *RunID* is NULL, the results from all run operations in TDSTATS.CommandsHistoryTbl will be shown.

## Output

| Column       | Description                                                                                                                                                                                                                                             |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Status       | Possible values: <ul style="list-style-type: none"> <li>• P (Pending)</li> <li>• I (In progress)</li> <li>• C (Completed successfully)</li> <li>• E (Error)</li> <li>• A (Aborted by the user)</li> </ul>                                               |
| PriorityRank | System assigned submission priority relative to other collection commands in the same run operation.                                                                                                                                                    |
| DatabaseName | Database on which the collection is performed.                                                                                                                                                                                                          |
| TableName    | Table on which the collection is performed.                                                                                                                                                                                                             |
| StartTime    | Start time for an in-progress or completed collections.                                                                                                                                                                                                 |
| EndTime      | End time for completed collections.                                                                                                                                                                                                                     |
| Skipped      | Possible values: <ul style="list-style-type: none"> <li>• Y means the statistics are submitted but not collected according to the defined THRESHOLD setting.</li> <li>• N means the statistics were collected and updated in the dictionary.</li> </ul> |

| Column      | Description                                                                              |
|-------------|------------------------------------------------------------------------------------------|
| RequestID   | System assigned ID for the COLLECT STATISTICS statement.                                 |
| CollectText | SQL text of the COLLECT STATISTICS statement.                                            |
| ErrorText   | Error message text if the Status output parameter returns an E. For details, see Status. |
| RunID       | System assigned ID of the RunCollect results reported.                                   |

## Usage Notes

To display results from one or more completed run operations, you must call `SelectStatsExecutionHistory`. For more information, see [SelectStatsExecutionHistory](#).

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Displaying Execution Results

| To display ...                                                                                   | You must specify ...                                                                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in-progress execution results or the most recent completed results for a specified commands list | a value other than NULL for either <i>CmdListID</i> or <i>CmdListName</i> . To identify a list by name, you must have assigned it during a prior call to <code>PrepCollect</code> .<br><br><b>Note:</b><br>You can specify <i>CmdListID</i> and <i>CmdListName</i> as long as they identify the same commands list. |
| a particular execution run operation that may not be the most recent                             | a value for <i>RunID</i> .<br><br><b>Note:</b><br>When you specify a value other than NULL for <i>RunID</i> , the values for <i>CmdListID</i> and <i>CmdListName</i> are not relevant and can remain NULL.                                                                                                          |

## Example: Using RunCollectReport

The following example shows how to display all pending statistics that are submitted but not collected according to the THRESHOLD setting.

```
CALL TDSTATS.RunCollectReport(null,'mycmds',1);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
```

```

*** ResultSet# 1 : 6 rows returned by "TDSTATS.RUNCOLLECTREPORT".
Status PriorityRank DataBaseName TableName StartTime
EndTime Skip Req. ID Collect Text
C 1 personnel charges 2013-02-14
18:25:08.790000+00:00 2013-02-14 18:25:08.850000+00:00 N 00007 COLLECT
STATISTICS INDEX("Proj_Id") ON "personnel"."charges";
C 2 personnel charges 2013-02-14 18:25:09.180000+00:00 2013-02-14
18:25:09.250000+00:00 N 100008 COLLECT STATISTICS INDEX("EmpNo","Proj_Id")
ON "personnel"."charges";
C 3 personnel department 2013-02-14
18:25:09.600000+00:00 2013-02-14 18:25:09.660000+00:00 N 100009 COLLECT
STATISTICS INDEX("DeptNo") ON "personnel"."department";
C 4 personnel employee 2013-02-14
18:25:09.910000+00:00 2013-02-14 18:25:09.970000+00:00 N 100010 COLLECT
STATISTICS INDEX("Name") ON "personnel"."employee";
C 5 personnel employee 2013-02-14
18:25:10.310000+00:00 2013-02-14 18:25:10.380000+00:00 N 100011 COLLECT
STATISTICS INDEX("EmpNo") ON "personnel"."employee";
C 6 personnel project 2013-02-14
18:25:10.720000+00:00 2013-02-14 18:25:10.810000+00:00 N 100012 COLLECT
STATISTICS INDEX("Proj_Id") ON "personnel"."project";

```

## Related Information

| For more information on ...                                                                             | See ...                                                                  |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| displaying results from one or more past executions with the option of specifying a variety of criteria | <a href="#">SelectStatsExecutionHistory</a> .                            |
| stored procedure dynamic result sets                                                                    | <i>Teradata Vantage™ - SQL External Routine Programming</i> , B035-1147. |

## SelectPreparedCollects

Displays the contents of a collection commands list generated by PrepCollect or collections for a specified object across all commands lists.

### Syntax

```

REPLACE PROCEDURE TDSTATS.SelectPreparedCollects (
  IN CmdListID      BIGINT,
  IN CmdListName    VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE
)
...
;

```

### Syntax Elements

#### CmdListID

System-assigned commands list ID from a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

### ***CmdListName***

User-assigned commands list name as specified during a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

### ***DatabaseName***

Name of the database. *DatabaseName* limits the results to collections for the database you specify.

### ***TableName***

Name of the table. *TableName* limits the results to collections on the table you specify.

### ***ObjectListName***

Name of object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose collections should be displayed.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

## **Output**

| Column       | Description                                                                                  |
|--------------|----------------------------------------------------------------------------------------------|
| CmdListID    | Commands list ID.                                                                            |
| CmdListName  | Command list name, if assigned.                                                              |
| DataBaseName | Database in which the collection is defined.                                                 |
| TableName    | Table on which the collection is defined.                                                    |
| PriorityRank | System assigned submission priority relative to other collections in the same commands list. |
| RequestID    | System assigned ID for the SQL COLLECT STATISTICS statement within the commands list.        |
| CollectText  | SQL text of the COLLECT STATISTICS statement.                                                |

## **Usage Notes**

### **Returning Result Set**

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Displaying the Collection Results

| To display the collection results for a ...                                                      | You must specify a ...                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| particular commands list                                                                         | non-value for <i>CmdListID</i> or <i>CmdListName</i> . You can specify both <i>CmdListID</i> and <i>CmdListName</i> as long as they identify the same command list.<br><br><b>Note:</b><br>If <i>CmdListID</i> and <i>CmdListName</i> are both NULL, the report will consider collections from all command lists in the TDSTATS database.                                                                 |
| particular object                                                                                | value for both <i>DatabaseName</i> or <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                                  |
| particular named object list that cannot be specified by <i>DatabaseName</i> or <i>TableName</i> | value for <i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.<br><br><b>Note:</b><br><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL. |

## Example: Using SelectPreparedCollects

The following example shows how to display the collection results for the commands list named MyCmdsList.

```
CALL TDSTATS.SelectPreparedCollects(NULL, 'MyCmdslist', NULL, NULL, NULL);
*** Query completed. 4 rows found. 7 columns returned.
*** Total elapsed time was 1 second.
```

| CmdListID | CmdListName | DatabaseName | TableName  | Priority | RequestID | CollectText          |
|-----------|-------------|--------------|------------|----------|-----------|----------------------|
| 5         | MyCmdsList  | Personnel    | Employee   | 1        | 34560     | COLLECT STATISTICS.. |
| 5         | MyCmdsList  | Personnel    | Department | 2        | 34561     | COLLECT STATISTICS.. |
| 5         | MyCmdsList  | Personnel    | Charges    | 3        | 34562     | COLLECT STATISTICS.. |

## Related Information

| For more information on ...               | See ...                                                                                  |
|-------------------------------------------|------------------------------------------------------------------------------------------|
| the PrepCollect external stored procedure | <a href="#">PrepCollect</a> .                                                            |
| SQL COLLECT STATISTICS statement          | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144. |

## SelectStatsExecutionHistory

Displays the collection execution history for a specified completed run, commands list, user object names, or time period.

### Syntax

```
REPLACE PROCEDURE TDSTATS.SelectStatsExecutionHistory (
  IN RunID          BIGINT,
  IN CmdListID     BIGINT,
  IN CmdListName   VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName  VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName     VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN StartTime     TIMESTAMP(6) WITH TIME ZONE,
  IN EndTime       TIMESTAMP(6) WITH TIME ZONE
)
...
;
```

### Syntax Elements

#### *RunID*

System assigned ID for the results from a given execution of RunCollect or RecollectTable.

To show the results from a specific execution run, you must specify a value for this input parameter.

RunID is an output parameter for RunCollect and ReCollectTable. If you specify NULL, the results from all run operations in the TDSTATS.CommandsHistoryTbl table will be displayed.

#### *CmdListID*

System-assigned commands list ID from a prior call to PrepCollect.

#### *CmdListName*

User-assigned commands list name.

#### *DatabaseName*

Name of the database. *DatabaseName* limits the results to collections for the database you specify.

#### *TableName*

Name of the table. *TableName* limits the results to collections on the table you specify.

**ObjectListName**

Name of object list. *ObjectListName* limits the results to completed collections on the object list (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) you specify.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

**StartTime**

Start time. *StartTime* limits the results to those collections submitted on or after the time you specify.

**EndTime**

End time. *EndTime* limits the results to those collections completed on or before the time you specify.

**Output**

| Column       | Description                                                                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RunID        | System assigned ID of the RunCollect results being reported.                                                                                                                                                                                             |
| CmdListID    | Commands list ID from the collection executed.                                                                                                                                                                                                           |
| CmdListName  | Commands list name, if assigned, from which the collection was executed from.                                                                                                                                                                            |
| StartTime    | Time when the collection was submitted for execution.                                                                                                                                                                                                    |
| EndTime      | Time when the collection completed execution.                                                                                                                                                                                                            |
| DatabaseName | Database on which the collection was performed.                                                                                                                                                                                                          |
| TableName    | Table on which the collection was performed.                                                                                                                                                                                                             |
| Status       | Possible values: <ul style="list-style-type: none"> <li>• C means the operation completed successfully.</li> <li>• E means an error in the operation occurred.</li> <li>• A means the operation was aborted by the user.</li> </ul>                      |
| Skipped      | Possible values: <ul style="list-style-type: none"> <li>• Y means the statistics were submitted but not collected according to the defined THRESHOLD setting.</li> <li>• N means the statistics were collected and updated in the dictionary.</li> </ul> |
| CollectText  | SQL text of the COLLECT STATISTICS statement.                                                                                                                                                                                                            |
| ErrorText    | Error message if the Status output parameter returns an E. For details, see the Status output parameter.                                                                                                                                                 |



## Usage Notes

The `SelectStatsExecutionHistory` external stored procedure displays the historical results from already completed calls to `RunCollect` or `RecollectTable`.

You can call `RunCollectReport` to monitor the results for an in-progress `RunCollect`.

## Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## Displaying Execution Results

| To display ...                                                            | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| results for a particular execution                                        | <i>RunID</i> .                                                                                                                                                                                                                                                                                                                                                                                                       |
| past execution results for a particular command list                      | <i>CmdListID</i> or <i>CmdListName</i> .                                                                                                                                                                                                                                                                                                                                                                             |
| past execution results from collections on a particular user object       | both <i>DatabaseName</i> or <i>TableName</i>                                                                                                                                                                                                                                                                                                                                                                         |
| past execution results from collections on a particular named object list | <p><i>ObjectListName</i> that was previously created by the <code>CreateObjectList</code> and <code>AddObjectListEntry</code> external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i>. When you specify a value other than NULL for <i>ObjectListName</i>, <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |
| execution results for a particular date range                             | both <i>StartTime</i> and <i>EndTime</i> .                                                                                                                                                                                                                                                                                                                                                                           |

## Example: Using SelectStatsExecutionHistory

The following example shows how to display the past execution results for the commands list named `MyCmdslist`.

```
CALL TDSTATS.SelectStatsExecutionHistory(NULL,NULL,'mycmds',NULL,NULL,NULL,NULL,NULL);
*** Procedure has been executed.
*** Warning: 3212 The stored procedure returned one or more result sets.
*** Total elapsed time was 1 second.
*** ResultSet# 1 : 6 rows returned by "TDSTATS.SELECTSTATSEXECUTIONHISTORY".
RunID CmdListID CmdListName  StartTime          EndTime
DataBaseName TableName Status Skipped      CollectText
-----
-----
```

```

-----
1      1      mycmds      2013-02-14 18:25:08.790000+00:00 2013-02-14
18:25:08.850000+00:00 personnel charges C N COLLECT STATISTICS
INDEX("Proj_Id") ON "personnel"."charges";
1      1      mycmds      2013-02-14 18:25:09.180000+00:00 2013-02-14
18:25:09.250000+00:00 personnel charges C N COLLECT STATISTICS
INDEX("EmpNo","Proj_Id") ON "personnel"."charges";
1      1      mycmds      2013-02-14 18:25:09.600000+00:00 2013-02-14
18:25:09.660000+00:00 personnel department C N COLLECT STATISTICS
INDEX("DeptNo") ON "personnel"."department";
1      1      mycmds      2013-02-14 18:25:09.910000+00:00 2013-02-14
18:25:09.970000+00:00 personnel employee C N COLLECT STATISTICS
INDEX("Name") ON "personnel"."employee";
1      1      mycmds      2013-02-14 18:25:10.310000+00:00 2013-02-14
18:25:10.380000+00:00 personnel employee C N COLLECT STATISTICS
INDEX("EmpNo") ON "personnel"."employee";
1      1      mycmds      2013-02-14 18:25:10.720000+00:00 2013-02-14
18:25:10.810000+00:00 personnel project C N COLLECT STATISTICS
INDEX("Proj_Id") ON "personnel"."project";

```

## Related Information

| For more information on ...                      | See ...                                                                                                                                                                                                                                                                |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| monitoring results for an in-progress RunCollect | <a href="#">RunCollectReport</a> .                                                                                                                                                                                                                                     |
| THRESHOLD settings                               | <ul style="list-style-type: none"> <li><a href="#">UpdateStatThresholdSetting</a>.</li> <li><i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i>, B035-1144.</li> <li><i>Teradata Vantage™ - Database Administration</i>, B035-1093.</li> </ul> |
| SQL COLLECT STATISTICS statements                | <i>Teradata Vantage™ - SQL Data Definition Language Syntax and Examples</i> , B035-1144.                                                                                                                                                                               |

## AbortCollect

Aborts one or more pending or in-progress collection statements within a running instance of RunCollect.

### Syntax

```

REPLACE PROCEDURE TDSTATS.AbortCollect (
  IN CmdListID BIGINT,
  IN CmdListName VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName VARCHAR(128) CHARACTER SET UNICODE,
  IN RequestID BIGINT,
  IN AbortInProgress CHAR(1) CHARACTER SET LATIN,
  OUT NumPendingAborted INTEGER,
  OUT NumInProgressAborted INTEGER
)

```

```
...  
;
```

## Syntax Elements

### ***CmdListID***

System-assigned commands list ID from a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

### ***CmdListName***

Commands list name executed by an in-progress RunCollect.

For more information about this input parameter, see the usage notes.

### ***DatabaseName***

Name of the database. *DatabaseName* identifies the database whose collections within the commands list you specify should be aborted.

### ***TableName***

Name of the table. *TableName* identifies the table whose collections within the commands list you specify should be aborted.

### ***RequestID***

Individual collection ID. *RequestID* identifies the individual collection ID within the commands list you specify that should be aborted.

### ***AbortInProgress***

Possible values:

- Y. If you specify Y, AbortCollect aborts the qualifying collections even if they are already in progress.
- N. If you specify N, AbortCollect allows in-progress collections to finish. This is the default value.

### ***NumPendingAborted***

Number of pending collection statements that were aborted.

### ***NumInProgressAborted***

Number of in-progress collection statements that were aborted.

## Usage Notes

### Identifying a Command List

#### Note:

You can specify both *CmdListID* and *CmdListName* as long as they identify the same commands list.

| To identify ...                       | You must ...                                                                         |
|---------------------------------------|--------------------------------------------------------------------------------------|
| the commands list being executed      | specify a value that is not null for either <i>CmdListID</i> or <i>CmdListName</i> . |
| a commands list by <i>CmdListName</i> | assign it during a prior call to <i>PrepCollect</i> .                                |

### Aborting Collection Statements

You must specify a *DatabaseName*, *TableName* or *RequestID* input parameter or all collections in the specified command list will be aborted. Those collection statements that do not qualify based on the input parameters you specified are unaffected and will be executed by *RunCollect*. For more information, see [RunCollect](#).

| To abort ...                                        | You must specify a value for ...                                                                                                                                                           |
|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| all collection statements for a particular database | <i>DatabaseName</i> .                                                                                                                                                                      |
| all collection statements for a particular table    | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                            |
| a single collection                                 | <i>RequestID</i> .<br><b>Note:</b><br>When specifying a <i>RequestID</i> , it is not necessary to specify <i>DatabaseName</i> or <i>TableName</i> since it is unique across all databases. |

### Example: Using AbortCollect

The following example shows how to abort all collections within commands list 1 that operate on the Personnel database.

```
CALL TDSTATS.AbortCollect (1,'MyCmdsList', 'PERSONNEL', NULL, NULL,
'Y', NumPendingAborted,NumInProgressAborted);
*** Total elapsed time was 1 second.
    NumPendingAborted    NumInProgressAborted
    -----
                2                1
```

## DBAControl Open APIs for Excluding Objects

Use these external stored procedures to permanently exclude certain objects and their defined statistics from the Automate, Analyzer, and Collect-related external stored procedures, where such exclusions supersede any inputs to those external stored procedures.

### AddExcludedObject

Specifies one or more objects whose statistics are to be permanently excluded from the Automate, Analyzer, and Collect-related operations.

#### Syntax

```
REPLACE PROCEDURE TDSTATS.AddExcludedObject (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName    VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumAdded    INTEGER
)
...
;
```

#### Syntax Elements

##### ***DatabaseName***

Name of the database. *DatabaseName* identifies the excluded database or the database containing the excluded table.

##### ***TableName***

Name of the table within the specified *DatabaseName*. *TableName* identifies the table to exclude within *DatabaseName*.

##### ***NumAdded***

Number of objects excluded.

#### Usage Notes

When you exclude objects by AddExcludedObject, it supersedes any argument values specified in calls to Automate, Analyze, and Collect-related APIs and allows Automate, Analyze, and Collect-related APIs to be called in an "everything but" manner.

Exclusion is not applicable to the TruncateAnalyzerHistory, TruncateAutomateHistory, and TruncateCollectHistory external stored procedures. Rows corresponding to any excluded objects are still deleted as part of the truncation operation assuming they qualify based on the scope of the specified

input parameters. For more information about these external stored procedures, see [DBAControl Open APIs for Managing Space](#).

Excluding Statistics

| To exclude all ...                                        | You must specify a value for ...           |
|-----------------------------------------------------------|--------------------------------------------|
| statistics for a particular table                         | <i>DatabaseName</i> and <i>TableName</i> . |
| objects and their statistics within a particular database | <i>DatabaseName</i> only.                  |

Example: Using AddExcludedObject

The following example shows how to exclude the table named Project and automate all statistics from the Personnel database, except those on the Project table.

```
CALL TDSTATS.AddExcludedObject('Personnel','Project', NumAdded);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumAdded
-----
      1
CALL TDSTATS.AutomateStats ('Personnel', NULL, NULL, NULL, 'Y', 'N', AutomateId,
NumCopied, NumRemoved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
AutomateId      NumCopied      NumRemoved
-----
      1              4              0
```

RemoveExcludedObject

Reinstates a previously excluded object.

Syntax

```
REPLACE PROCEDURE TDSTATS.RemoveExcludedObject (
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName    VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumRemoved  INTEGER
)
...
;
```

Syntax Elements

DatabaseName

Name of the database. *DatabaseName* allows you to specify the database to reinstate or the database containing a table to reinstate.

TableName

Name of the table within the specified *DatabaseName*. *TableName* allows you to specify a table within *DatabaseName* to be reinstated.

NumRemoved

Number of objects removed from the exclusion list.

Usage Notes

Reinstated objects and their statistics are considered by subsequent invocations of Automate, Analyzer, and Collect-related external stored procedures.

Reinstating Statistics

| To reinstate all ...                               | You must specify a value for ...                |
|----------------------------------------------------|-------------------------------------------------|
| objects and statistics within a specified database | <i>DatabaseName</i> only.                       |
| statistics for a specified table                   | both <i>DatabaseName</i> and <i>TableName</i> . |

Example: Using RemoveExcludedObject

The following example shows how to reinstate the Personnel.Project table and prepare all statistics for the Personnel database, including those on the Personnel.Project table.

```
CALL TDSTATS.RemoveExcludedObject('PERSONNEL','PROJECT', NumRemoved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
  NumRemoved
  -----
         1
CALL TDSTATS.PrepareCollect ('PERSONNEL', NULL, NULL, NULL, NumStmtsPrepped,
NumStatsPrepped, CmdListID);
*** Procedure has been executed.
*** Total elapsed time was 3 seconds.
```

## SelectAllExcludedObjects

Displays the list of all user objects whose collections are explicitly excluded from Automated Statistics Management.

### Syntax

```
REPLACE PROCEDURE TDSTATS.SelectAllExcludedObjects ()
    ...
;
```

### Output

| Column           | Description                                                                                                       |
|------------------|-------------------------------------------------------------------------------------------------------------------|
| ObjectType       | Type or level of exclusion: <ul style="list-style-type: none"> <li>• D (Database)</li> <li>• T (Table)</li> </ul> |
| DatabaseName     | Excluded database or the database containing the excluded table.                                                  |
| TableName        | Table to be excluded if ObjectType is T. For details, see the ObjectType output parameter.                        |
| CreatedTimeStamp | Time when the object was excluded.                                                                                |
| CreateUser       | User who imposed the exclusion.                                                                                   |

### Usage Notes

#### Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

#### Example: Using SelectAllExcludedObjects

The following example shows how to retrieve a report of all excluded objects defined for the Automated Statistics Management feature.

```
CALL TDSTATS.SelectAllExcludedObjects;
*** Query completed. 3 rows found. 6 columns returned.
*** Total elapsed time was 1 second.
ObjectType DatabaseName TableName CreatedTimeStamp CreateUser
```



|   |           |            |                         |
|---|-----------|------------|-------------------------|
| T | PERSONNEL | PROJECT    | 2010-09-01 23:46:29 DBC |
| T | PERSONNEL | DEPARTMENT | 2010-09-01 23:48:34 DBC |
| T | SALES     | SUPPLIER   | 2010-09-05 21:31:35     |

## DBAControl Open APIs for Statistic Settings

In cases where you need to override the default system settings, use the following external stored procedures to control the various syntax options used in the preparation of the SQL COLLECT STATISTICS statements.

The new system settings applied will take effect during the next RunCollect involving the selected statistics. For more information about this external stored procedure, see [RunCollect](#).

## UpdateStatHistogramSettings

Records the user-specified MAXINTERVAL or MAXVALUELENGTH setting to include when preparing and executing the SQL COLLECT STATISTICS statements on selected statistics or objects.

### Syntax

```

REPLACE PROCEDURE TDSTATS.UpdateStatHistogramSettings (
  IN SCOID BIGINT,
  IN DatabaseName      VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName         VARCHAR(128) CHARACTER SET UNICODE,
  IN MaxIntervals      INTEGER,
  IN MaxValueLength    INTEGER,
  OUT NumUpdated       INTEGER
)
...
;

```

### Syntax Elements

#### SCOID

TDSTATS ID of an individual statistic whose histogram settings are to be changed.

#### DatabaseName

Name of the database. *DatabaseName* limits the new settings applied to all statistics collected within the database you specify.

**TableName**

Name of the table within the specified *DatabaseName*. *TableName* limits the new settings to statistics collected on the table you specify within *DatabaseName*.

**MaxIntervals**

Maximum number of histogram intervals used for collected statistics.

A value of -1 indicates the SYSTEM selected value is used. For more information, see the COLLECT STATISTICS MAXINTERVALS option in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

**MaxValueLength**

Maximum size for histogram values.

A value of -1 indicates the SYSTEM selected value is used. For information on the valid range of values, see the COLLECT STATISTICS MAXVALUELENGTH option in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

**NumUpdated**

Number of statistics whose histogram settings have been updated.

**Usage Notes**

The new collection settings will take effect during the next RunCollect operation involving the selected statistics. For more information, see [RunCollect](#).

**Updating Statistics**

| To update ...                           | You must specify a value for ...                                              |
|-----------------------------------------|-------------------------------------------------------------------------------|
| a single statistics collection          | <i>SCOID</i> .<br><b>Note:</b><br>This input parameter cannot be NULL.        |
| all statistics for a specified database | <i>DatabaseName</i> .<br><b>Note:</b><br>This input parameter cannot be NULL. |
| all statistics for a specified table    | both <i>DatabaseName</i> and <i>TableName</i> .                               |

**Example: Using UpdateStatHistogramSettings**

The following example shows how to specify a MAXINTERVALS value of 500 when collecting statistics on the individual statistic whose TDSTATS database ID is 102.

```
CALL TDSTATS.UpdateStatHistogramSettings(102, 'Personnel', 'Employee', 500,
0, NumUpdated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumUpdated
-----
1
```

## UpdateStatSampleSetting

Records the user specified SAMPLE setting to include when preparing and executing the SQL COLLECT STATISTICS statements on the selected statistics or objects.

### Syntax

```
REPLACE PROCEDURE TDSTATS.UpdateStatSampleSetting (
  IN  SCOID          BIGINT,
  IN  DatabaseName  VARCHAR(128) CHARACTER SET UNICODE,
  IN  TableName     VARCHAR(128) CHARACTER SET UNICODE,
  IN  SampleType    CHAR(1) CHARACTER SET LATIN,
  IN  Percentage    INTEGER,
  OUT NumUpdated    INTEGER
)
...
;
```

### Syntax Elements

#### SCOID

TDSTATS ID of an individual statistic whose SAMPLE settings are to be changed. For more information, see the *SampleType* input parameter.

#### DatabaseName

Name of the database. *DatabaseName* limits the new settings that will be applied to all statistics collected within the database you specify.

#### TableName

Name of the table within the specified *DatabaseName*. *TableName* limits the new settings to statistics collected on the table you specify within *DatabaseName*.

#### SampleType

Type of sample option change:

- S means SYSTEM SAMPLE
- U means SAMPLE <Percentage> PERCENT
- N means no sampling

For more information about the COLLECT STATISTICS SAMPLE option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

### Percentage

User supplied percentage.

This value is valid when *SampleType* is U. For details, see the *SampleType* input parameter.

### NumUpdated

Number of statistics whose SAMPLE setting have been updated.

For more information about the COLLECT STATISTICS SAMPLE option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## Usage Notes

The new collection settings will take effect during the next RunCollect involving the selected statistics. For more information about this external stored procedure, see [RunCollect](#).

### Example: Using UpdateStatSampleSetting

The following example shows how to specify a *SampleType* option (for example, SAMPLE 25 PERCENT) when collecting statistics on the individual statistic whose TDSTATS database ID is 102 on the Personnel.Employee table.

```
CALL TDSTATS.UpdateStatSamplingSetting(102, 'Personnel', 'Employee', 'U',
25, NumUpdated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumUpdated
-----
1
```

## Related Information

For more information about the COLLECT STATISTICS SAMPLE option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

## UpdateStatThresholdSetting

Records the user-specified THRESHOLD settings to include when preparing and executing SQL COLLECT STATISTICS statements on selected statistics or objects.

### Syntax

```
REPLACE PROCEDURE TDSTATS.UpdateStatThresholdSetting (
  IN  SCOID          BIGINT,
  IN  DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN  TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN  ThresholdType   CHAR(1) CHARACTER SET LATIN,
  IN  AgeThreshold    INTEGER,
  IN  GrowthThreshold INTEGER,
  IN  ForCurrentlyOnly CHAR(1) CHARACTER SET LATIN,
  OUT NumUpdated      INTEGER
)
...
;
```

### Syntax Elements

#### *SCOID*

TDSTATS ID of an individual statistic whose THRESHOLD settings are to be changed.  
This input parameter cannot be NULL.

#### *DatabaseName*

*DatabaseName* limits the new settings that will be applied to all statistics collected within the database you specify.

This input parameter cannot be NULL.

#### *TableName*

Name of the table within the specified *DatabaseName*. *TableName* limits the new settings to statistics collected on the table you specify within *DatabaseName*.

#### *ThresholdType*

Type of threshold option change:

- S means SYSTEM THRESHOLD
- U means THRESHOLD <*AgeThreshold*> DAYS AND THRESHOLD <*GrowthThreshold*> PERCENT

- N means no threshold.

When specifying the U option, you must also specify a value other than NULL for the *AgeThreshold* or *GrowthThreshold* input parameter, or both.

For more information about the COLLECT STATISTICS THRESHOLD option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144 or *Teradata Vantage™ - Database Administration*, B035-1093.

### ***AgeThreshold***

User-supplied age threshold in number of days.

This input parameter is valid only when *SampleType* is U. For more information about the *SampleType* input parameter, see [UpdateStatSampleSetting](#).

A NULL or zero value indicates a no age threshold.

### ***GrowthThreshold***

User-supplied growth threshold as a percentage.

This input parameter is valid only when *SampleType* is U. For more information about the *SampleType* input parameter, see [UpdateStatSampleSetting](#).

A NULL or zero value indicates a no growth threshold.

### ***ForCurrentlyOnly***

Possible values:

- Y. If you specify Y, the THRESHOLD setting will be issued along with the FOR CURRENT option making it applicable to the current collection only.
- N or NULL. If you specify N or NULL, the THRESHOLD setting is applied to all future collections.

For more information about the THRESHOLD FOR CURRENT option, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

### ***NumUpdated***

Number of statistics whose THRESHOLD settings have been updated.

For more information about the THRESHOLD option, see SQL COLLECT STATISTICS in *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144 or *Teradata Vantage™ - Database Administration*, B035-1093.

## **Usage Notes**

The new collection settings will take effect during the next RunCollect operation involving the selected statistics. For more information, see [RunCollect](#).

## Updating Statistics

| To update ...                            | You must specify a value for ...                                                                                                                                                                   |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single statistics collection           | <i>SCOID</i> .<br><b>Note:</b><br>This input parameter cannot be NULL. When specifying a value for <i>SCOID</i> , you do not need to specify a value for <i>DatabaseName</i> or <i>TableName</i> . |
| all statistics for a particular database | <i>DatabaseName</i> .<br><b>Note:</b><br>This input parameter cannot be NULL.                                                                                                                      |
| all statistics for a particular table    | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                    |

### Example: Using UpdateStatThresholdSetting

The following example shows how to specify a *ThresholdType* option (for example, THRESHOLD 10 PERCENT AND THRESHOLD 7 DAYS) when collecting statistics on the individual statistic whose TDSTATS database ID is 103.

```
CALL TDSTATS.UpdateStatThresholdSetting(103, 'Personnel', 'Employee', 'U', 7,
10, NULL, NumUpdated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumUpdated
-----
1
```

## DBAControl Open APIs for Prepared Collections

Use these external stored procedures to modify and customize the prepared list of collection commands.

### Note:

Actions taken by the DBAControl external stored procedures for Prepared Collections are relevant only to a specified prepared list and future invocations of PrepCollect will not retain them. For more information, see [PrepCollect](#).

## AddPreparedCollect

Adds a user specified COLLECT STATISTICS statement to an already prepared list of collections generated by PrepCollect.

## Syntax

```
REPLACE PROCEDURE TDSTATS.AddPreparedCollect (
  IN  CmdListID    BIGINT,
  IN  CmdListName  VARCHAR(128) CHARACTER SET UNICODE,
  IN  Priority      INTEGER,
  IN  StatsType    CHAR(1) CHARACTER SET LATIN,
  IN  DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN  TableName    VARCHAR(128) CHARACTER SET UNICODE,
  IN  CollectText  VARCHAR(10000) CHARACTER SET UNICODE,
  OUT NumAdded     INTEGER
)
...
;
```

## Syntax Elements

### ***CmdListID***

Commands list ID as assigned from prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

### ***CmdListName***

User-named commands list as specified in a prior call to PrepCollect.

For more information about this input parameter, see the usage notes.

### ***Priority***

Priority of the added collection. The value is expressed as a relative rank among other collections in the same commands list.

To determine a *Priority* value, you can call the SelectPreparedCollects external stored procedure. For more information, see [SelectPreparedCollects](#).

### ***StatsType***

This input parameter is reserved for future use.

### ***DatabaseName***

Name of the database containing the statistic to be collected.

This input parameter does not permit wildcard characters.



**TableName**

Name of the table on whose columns statistics are collected.

This input parameter does not permit wildcard characters.

**CollectText**

User-specified COLLECT STATISTICS statement text.

**NumAdded**

Possible values:

- 1 means the operation was successful.
- 0 means the operation failed.

**Usage Notes**

You can use the AddPreparedCollect external stored procedure to customize a system generated commands list with additional collections.

The added collection is only stored in the specified commands list. Future invocations of PrepCollect will not incorporate the added collection in its generated commands lists.

To permanently automate the collection, you must call AutomateSingleStat or AutomateStats.

**Identifying a Command List****Note:**

You can specify both *CmdListID* and *CmdListName* as long as they identify the same commands list.

| To identify ...                       | You must ...                                                                         |
|---------------------------------------|--------------------------------------------------------------------------------------|
| the commands list being executed      | specify a value that is not null for either <i>CmdListID</i> or <i>CmdListName</i> . |
| a commands list by <i>CmdListName</i> | assign it during a prior call to PrepCollect.                                        |

**Collecting Statistics**

To collect statistics for a particular object, you must specify a value for both *DatabaseName* and *TableName* in case the RemovedPreparedCollect external stored procedure is subsequently called.

**Note:**

*DatabaseName* and *TableName* do not permit wildcard characters.

Example: Using AddPreparedCollect

The following example shows how to add a collection on the Personnel.JobTitle table to the assigned number command list ID generated by PrepCollect (for example, 3).

```
CALL TDSTATS.AddPreparedCollect(3, NULL, 1, NULL, 'Personnel', 'Employee,
'Collect Statistics Column(JobTitle) On Personnel.Employee', NumAdded);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumAdded
-----
          1
```

Related Information

| For more information ...              | See ...                                                                                                                       |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| automating the collection permanently | <ul style="list-style-type: none"><li>• <a href="#">AutomateSingleStat</a></li><li>• <a href="#">AutomateStats</a>.</li></ul> |
| on removing one or more collections   | <a href="#">RemovePreparedCollect</a> .                                                                                       |

RemovePreparedCollect

Removes one or more collections from an already prepared commands list generated by PrepCollect or customizes a system-generated commands list.

Syntax

```
REPLACE PROCEDURE TDSTATS.RemovePreparedCollect (
  IN RequestID      BIGINT,
  IN CmdListID      BIGINT,
  IN CmdListName    VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumRemoved    INTEGER
)
...
;
```

Syntax Elements

**RequestID**  
TDSTATS ID for a prepared collection command.

**CmdListID**

Commands list ID as assigned from prior call to PrepCollect. For more information, see [PrepCollect](#).

**CmdListName**

User-named commands list as specified in prior call to PrepCollect. For more information, see [PrepCollect](#).

**DatabaseName**

Name of the database. If you specify a value other than NULL, *DatabaseName* removes all collections in the commands list that are defined on the database you specify.

This input parameter cannot be NULL.

**TableName**

Name of the table. If you specify a value other than NULL, *TableName* removes all collections in the commands list that are defined on the table you specify within *DatabaseName*.

This input parameter cannot be NULL.

**NumRemoved**

Number of collections removed.

**Usage Notes**

The qualifying collections are only removed from the specified commands list. Future invocations of PrepCollect may continue to incorporate the removed collection into newly generated commands lists.

To permanently remove automated collections, you must call the DeAutomateStats external stored procedure For more information, see [DeAutomateStats](#).

**Removing Collections**

| To remove ...       | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| a single collection | <i>RequestID</i> .<br><b>Note:</b><br>If you specify a value for <i>RequestID</i> , you do not need to specify values for <i>CmdsListId</i> , <i>DatabaseName</i> , or <i>ObjectName</i> .<br>If you do not specify a <i>RequestID</i> , you must specify a value other than NULL for <i>CmdsListId</i> or <i>CmdListName</i> to limit the removal to a particular prepared list of collection commands.<br>To determine the assigned <i>RequestID</i> for a particular collection, you can call <i>SelectPreparedCollects</i> . For more information, see <a href="#">SelectPreparedCollects</a> . |

| To remove ...                                                                | You must specify a value for ...                |
|------------------------------------------------------------------------------|-------------------------------------------------|
| all collections for a particular database within the specified commands list | <i>DatabaseName</i> .                           |
| all collections for a particular table within the specified command list     | both <i>DatabaseName</i> and <i>TableName</i> . |

### Example: Using RemovePreparedCollect

The following example shows how to remove all collections on the Personnel.Department table within the command list named MyCmdsList.

```
CALL
TDSTATS.RemovePreparedCollect(NULL,NULL,'MyCmdsList','PERSONNEL','Department',
NumRemoved);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
NumRemoved
-----
                2
```

## SelectPreparedCollects

Displays the contents of a collection commands list generated by PrepCollect or a specified object across all commands lists.

### Syntax

```
REPLACE PROCEDURE TDSTATS.SelectPreparedCollects (
  IN CmdListID      BIGINT,
  IN CmdListName    VARCHAR(128) CHARACTER SET UNICODE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE
)
...
;
```

### Syntax Elements

#### CmdListID

System assigned commands list ID from a prior call to PrepCollect. For more information, see [PrepCollect](#).

For more information about this input parameter, see the usage notes.

### ***CmdListName***

User-assigned commands list name as specified during prior call to `PrepCollect`. For more information, see [PrepCollect](#).

For more information about this input parameter, see the usage notes.

### ***DatabaseName***

Name of the database. *DatabaseName* limits the results to collections on the database you specify.

### ***TableName***

Name of the table. *TableName* limits the results to collections on the table you specify.

### ***ObjectListName***

Name of the object list. *ObjectListName* allows you to specify a previously defined list of objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) whose collections should be displayed.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

## **Output**

| Column       | Description                                                                                  |
|--------------|----------------------------------------------------------------------------------------------|
| CmdListID    | Commands list ID.                                                                            |
| CmdListName  | Command list name, if assigned.                                                              |
| DataBaseName | Database in which the collection is defined.                                                 |
| TableName    | Table on which the collection is defined.                                                    |
| PriorityRank | System assigned submission priority relative to other collections in the same commands list. |
| RequestID    | System assigned ID for the COLLECT STATISTICS statement within the commands list.            |
| CollectText  | SQL text of the COLLECT STATISTICS statement.                                                |

## Usage Notes

### Returning Result Set

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

### Specifying Collection Statements to Display

| To display ...                                 | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| collections from a particular command list     | <p>either <i>CmdListID</i> or <i>CmdListName</i>.<br/>To identify a command list by name, you must have assigned it during a prior call to <i>PrepCollect</i>.</p> <p><b>Note:</b><br/>You can specify both <i>CmdListID</i> and <i>CmdListName</i> as long as they identify the same commands list.<br/>If <i>CmdListID</i> and <i>CmdListName</i> are both NULL, the report will return collections from all command lists in the TDSTATS database.</p> |
| collections on a particular object             | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                                                                                           |
| collections for a particular named object list | <p><i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.</p> <p><b>Note:</b><br/><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i>. When you specify a value other than NULL for <i>ObjectListName</i>, <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p>                                                     |

### Example: Using SelectPreparedCollects

The following example shows how to display the collection commands in the user assigned command list named *MyCmdsList*.

```
CALL TDSTATS.SelectPreparedCollects(NULL, 'MyCmdslist', NULL, NULL, NULL);
*** Query completed. 4 rows found. 7 columns returned.
*** Total elapsed time was 1 second.
```

| CmdListID | CmdListName | DatabaseName | TableName  | Priority | RequestID | CollectText          |
|-----------|-------------|--------------|------------|----------|-----------|----------------------|
| 5         | MyCmdsList  | Personnel    | Employee   | 1        | 34560     | COLLECT STATISTICS.. |
| 5         | MyCmdsList  | Personnel    | Department | 2        | 34561     | COLLECT STATISTICS.. |
| 5         | MyCmdsList  | Personnel    | Charges    | 3        | 34562     | COLLECT STATISTICS.. |

## PrioritizePreparedCollect

Changes the submission order of an individual collection within a prepared commands list.

### Syntax

```
REPLACE PROCEDURE TDSTATS.PrioritizePreparedCollect (
  IN RequestID    BIGINT,
  IN Priority     INTEGER
  OUT NumUpdated INTEGER
)
...
;
```

### Syntax Elements

#### *RequestID*

TDSTATS database unique ID for an individual prepared collection.

This input parameter cannot be NULL.

#### *Priority*

Changed priority of the specified collection expressed as a relative rank among other collections in the commands list containing *RequestID*.

This input parameter cannot be NULL.

#### *NumUpdated*

Possible values:

- 1 means the operation was successful.
- 0 means the operation failed.

### Usage Notes

You can call the `SelectPreparedCollects` external stored procedure to determine the priority of other collections in the same commands list.

The modified priority is only relevant to the commands list containing the specified collection as identified by *RequestID*. Future invocations of `PrepCollect` will not incorporate the modified priority.

If two or more collections have the same priority within a commands list, the submission order among them is undefined.

**Example: Using PrioritizePreparedCollect**

The following example shows how to change the priority of pending collection number from 12 to 1 (that is, one being the highest priority).

```
CALL TDSTATS.PrioritizePreparedCollect(12, 1, NumUpdated);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
  NumUpdated
  -----
           1
```

**Related Information**

| For more information on ...                                  | See ...                                 |
|--------------------------------------------------------------|-----------------------------------------|
| the priority of other collections in the commands list       | <a href="#">SelectPreparedCollects.</a> |
| preparing a prioritize list of COLLECT STATISTICS statements | <a href="#">PrepCollect.</a>            |

# DBAControl Open APIs for Managing Space

Use these external stored procedures to manage space in the TDSTATS database.

## TruncateAnalyzerHistory

Truncates rows from the TDSTATS.AnalyzerHistoryTbl table.

**Syntax**

```
REPLACE PROCEDURE TDSTATS.TruncateAnalyzerHistory (
  IN OlderThan      TIMESTAMP(6) WITH TIME ZONE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN AnalysisId     BIGINT,
  OUT DeletedRows   INTEGER
)
...
;
```



## Syntax Elements

### ***OlderThan***

Time. *OlderThan* deletes all results older than the time you specify.

### ***DatabaseName***

Name of the database. *DatabaseName* deletes rows associated to the database you specify.

### ***TableName***

Name of the table within the specified *DatabaseName*. *TableName* limits the deletion to rows associated to the table you specify within *DatabaseName*.

### ***ObjectListName***

Name of the object list. *ObjectListName* deletes the rows associated with the object list (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) you specify.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***AnalysisId***

Name of the object list. *ObjectListName* deletes the rows associated with the object list (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) you specify.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***DeletedRows***

Number of deleted rows.

## Usage Notes

Before removing the contents of the TDSTATS.AnalyzerHistoryTb table, Teradata recommends that you review certain information, such as missing statistics recommendations, or it may be lost.

You must periodically run TruncateAnalyzerHistory to prevent the TDSTATS.AnalyzerHistoryTbl table from consuming an excessive amount of space.

## Removing Results

If you specify NULL for one of the following input parameters, all rows will be truncated.

| To remove results...                                                                      | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| from a particular Analyzer-related call or results that are older than the specified time | <i>AnalysisId</i> or <i>OlderThan</i> .                                                                                                                                                                                                                                                                                                                                                                    |
| on a particular object                                                                    | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                                            |
| for a particular named object list                                                        | <p><i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |

### Example: Using TruncateAnalyzerHistory

The following example shows how to remove all Analyzer-related results that are older than one year.

```
CALL TDSTATS.TruncateAnalyzerHistory (CAST(CURRENT_DATE - INTERVAL '1' YEAR AS
TIMESTAMP(6)),NULL, NULL, NULL, NULL, DeletedRows);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
DeletedRows
-----
          2450
```

## TruncateAutomateHistory

Truncates rows from the TDSTATS.AutomateHistoryTbl table.

### Syntax

```
REPLACE PROCEDURE TDSTATS.TruncateAutomateHistory (
  IN OlderThan      TIMESTAMP(6) WITH TIME ZONE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN AutomateId     BIGINT,
  OUT DeletedRows   INTEGER
)
...
;
```

## Syntax Elements

### ***OlderThan***

Time. *OlderThan* deletes the automate history data older than the time you specify.

If you specify NULL, time is not used as a qualifier for deletion.

### ***DatabaseName***

Name of the database. *DatabaseName* limits the deletion to rows associated with the database you specify.

### ***TableName***

Name of the table within the specified *DatabaseName*. *TableName* limits the deletion to rows associated to the table you specify within *DatabaseName*.

### ***ObjectListName***

Name of the object list. *ObjectListName* deletes the rows associated with the objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) in the list you specify.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***AutomateId***

Automate results ID. *AutomateId* deletes results associated with the Automate results ID you specify.

### ***DeletedRows***

Number of deleted rows.

## Usage Notes

You must periodically run `TruncateAutomateHistory` to prevent the `TDSTATS.AutomateHistoryTbl` table from consuming an excessive amount of space.

## Specifying the Results to Remove

If you specify NULL for one of the following input parameters, all rows will be truncated.

| To remove results ...                                                                 | You must specify a value for ...                |
|---------------------------------------------------------------------------------------|-------------------------------------------------|
| from a particular Automate-related call or results that older than the specified time | <i>AutomateId</i> or <i>OlderThan</i> .         |
| on a particular object                                                                | both <i>DatabaseName</i> and <i>TableName</i> . |

| To remove results ...              | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| for a particular named object list | <p><i>ObjectListName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectListName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectListName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |

### Example: Using TruncateAutomateHistory

The following example shows how to remove all Automate-related results that are older than one year.

```
CALL TDSTATS.TruncateAutomateHistory (CAST(CURRENT_DATE - INTERVAL '1' YEAR AS
TIMESTAMP(6)), NULL, NULL, NULL,NULL, DeletedRows);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
DeletedRows
-----
          300
```

## TruncateCollectHistory

Truncates rows from the TDSTATS.CommandsHistoryTbl table representing execution results from collection commands.

### Syntax

```
REPLACE PROCEDURE TDSTATS.TruncateCollectHistory (
  IN OlderThan      TIMESTAMP(6) WITH TIME ZONE,
  IN DatabaseName   VARCHAR(128) CHARACTER SET UNICODE,
  IN TableName      VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN RunId          BIGINT,
  IN CmdListID      BIGINT,
  IN CmdListName    VARCHAR(128) CHARACTER SET UNICODE,
  OUT DeletedRows   INTEGER
)
...
;
```

## Syntax Elements

### ***OlderThan***

Time. *OlderThan* deletes collection history data older than the time you specify.

### ***DatabaseName***

Name of the database. *DatabaseName* limits the deletion to rows associated with the database you specify.

### ***TableName***

Name of the table within the specified *DatabaseName*. *TableName* limits the deletion to rows associated with the table you specify within *DatabaseName*.

### ***ObjectListName***

Name of the object list. *ObjectListName* deletes rows associated with objects (that is, one or more database names or fully qualified table names where each name entry may optionally contain wildcard characters) in the list you specify.

For more information about this input parameter, see the usage notes or [CreateObjectList](#).

### ***RunId***

RunCollect results ID. *RunId* deletes history rows associated with the RunCollect results ID you specify.

### ***CmdListId***

Commands list ID. *CmdListId* deletes history rows associated with the commands list ID you specify.

### ***CmdListName***

Name of the commands list. *CmdListName* deletes history rows associated with the commands list you specify.

### ***DeletedRows***

Number of deleted history rows.

## Usage Notes

Before using `TruncateCollectHistory`, you should consider the potential impact of losing historical execution data.

You must periodically run `TruncateCollectHistory` to prevent the `TDSTATS.CommandsHistoryTbl` table from consuming an excessive amount of space.

## Removing Results

If you specify NULL for one of the following input parameters, all rows will be removed.

| To remove results ...                                                                    | You must specify a value for ...                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| from a particular collection run operation or results that older than the specified time | <i>RunID</i> or <i>OlderThan</i> .                                                                                                                                                                                                                                                                                                                                                             |
| on a particular object                                                                   | both <i>DatabaseName</i> and <i>TableName</i> .                                                                                                                                                                                                                                                                                                                                                |
| for a particular named object list                                                       | <p><i>ObjectName</i> that was previously created by the <i>CreateObjectList</i> and <i>AddObjectListEntry</i> external stored procedures.</p> <p><b>Note:</b></p> <p><i>ObjectName</i> usage is mutually exclusive with <i>DatabaseName</i> and <i>TableName</i> . When you specify a value other than NULL for <i>ObjectName</i> , <i>DatabaseName</i> and <i>TableName</i> must be NULL.</p> |

### Example: Using TruncateCollectHistory

The following example shows how to remove all Collect-related results that are older than one year.

```
CALL TDSTATS.TruncateCollectHistory (CAST(CURRENT_DATE - INTERVAL '1' YEAR AS
TIMESTAMP(6)),NULL, NULL, NULL, NULL, NULL, NULL, DeletedRows);
*** Procedure has been executed.
*** Total elapsed time was 1 second.
DeletedRows
-----
          2450
```

## Open APIs for Object Lists

Use these external stored procedures to create a named object list that represents any arbitrary set of databases or tables.

### CreateObjectList

Creates a new object list with a user-specified name.

#### Syntax

```
REPLACE PROCEDURE TDSTATS.CreateObjectList (
  IN ObjectName VARCHAR(128) CHARACTER SET UNICODE,
  OUT ObjectListId BIGINT
```

```
)
...
;
```

## Syntax Elements

### *ObjectListName*

Name for the newly created object list.

The name must be unique among all object lists stored in the TDSTATS database.

### *ObjectListId*

ID of the created object list.

## Usage Notes

After a new object list is created, you can call `AddObjectListEntry` to add objects to a list.

To add database or tables to the list, see [AddObjectListEntry](#).

### Example: Using `CreateObjectList`

The following example shows how to create a list consisting of all finance databases and cash tables within account databases.

```
CALL TDSTATS.CreateObjectList('FinCashList',ObjectListId);
CALL TDSTATS.AddObjectListEntry('FinCashList',NULL,'Fin%',NULL);
CALL TDSTATS.AddObjectListEntry('FinCashList',NULL,'Account%', 'Cash%');
```

## AddObjectListEntry

Adds a database or table name to an existing named object list.

### Syntax

```
REPLACE PROCEDURE TDSTATS.AddObjectListEntry (
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListId BIGINT,
  IN DatabaseName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectName VARCHAR(128) CHARACTER SET UNICODE
)
...
;
```

## Syntax Elements

### ***ObjectListName***

Name of the existing object list.

An object list may contain a mixture of databases and individual tables.

### ***ObjectListId***

ID of the existing object list.

### ***DatabaseName***

Name of the database containing the objects being added.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

This input parameter cannot be NULL.

### ***ObjectName***

Name of the table to be added.

If you specify NULL, all objects defined in the *DatabaseName* are added to the list.

If you specify a value other than NULL, you must define the *ObjectName* within *DatabaseName*.

This input parameter may contain wildcard characters (% , \_). For more information about wildcard characters, see the usage notes.

## Usage Notes

You must specify *ObjectListName* or *ObjectListId* to identify the list previously created via a call to `CreateObjectList`. For details, see [CreateObjectList](#).

Object list data can be viewed in the `TDSTATS.ObjectList` and `TDSTATS.ObjectListEntry` tables. The names of the object lists are stored in the `TDSTATS.ObjectList.ListName` column.

The SQL `DROP DATABASE`, `DROP TABLE`, `MODIFY DATABASE`, and `ALTER TABLE` statements will not have their actions automatically reflected in the `TDSTATS.ObjectListEntry` table. For more information about these SQL statements, see *Teradata Vantage™ - SQL Data Definition Language Syntax and Examples*, B035-1144.

For *DatabaseName* and *ObjectName*, wildcard characters are resolved dynamically at the time in which an object list is passed in calling the `Automate`, `Analyze`, or `PrepCollect` APIs. For more information about wildcard characters, see the following topic.



## Wildcard Characters

The following table describes the wildcard characters that the names of some databases, tables, and objects contain.

| Character        | Description                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| % (PERCENT SIGN) | Represents any string of zero or more arbitrary characters. Any string of characters is acceptable as a replacement for the percent.      |
| _ (LOW LINE)     | Represents exactly one arbitrary character. Any single character is acceptable in the position in which the underscore character appears. |

For more information on how to use these wildcard characters, see the SQL LIKE clause in *Teradata Vantage™ - SQL Functions, Expressions, and Predicates*, B035-1145.

### Example: Using AddObjectListEntry

The following example shows how to create an object list named MyObjects consisting of databases db1, db2.tx, and db3.t%.

```
CALL TDSTATS.CreateObjectList('MyObjects',ObjectListId);
CALL TDSTATS.AddObjectListEntry('MyObjects',NULL,'db1',NULL);
CALL TDSTATS.AddObjectListEntry('MyObjects',NULL,'db2','tx');
CALL TDSTATS.AddObjectListEntry('MyObjects',NULL,'db3','t%');
```

## RemoveObjectList

Removes a named object list and all of its entries.

### Syntax

```
REPLACE PROCEDURE TDSTATS.RemoveObjectList (
  IN ObjectListName VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListId BIGINT,
  OUT NumRemoved INTEGER
)
...
;
```

### Syntax Elements

#### *ObjectListName*

Name of the existing object list. This list may contain both databases and fully qualified tables.

This input parameter cannot be NULL.

### ***ObjectListId***

ID of the existing object list to remove.

This input parameter cannot be NULL.

### ***NumRemoved***

Number of the database and table entries that were in the removed list.

## **Usage Notes**

You can view object list data in the TDSTATS.ObjectList and TDSTATS.ObjectListEntry tables. The names of the object lists are stored in the TDSTATS.ObjectList.ListName column.

### **Example: Using RemoveObjectList**

The following example shows how to remove the newly created object list named FinCashList and its entries.

```
CALL TDSTATS.CreateObjectList('FinCashList',ObjectListId);
CALL TDSTATS.AddObjectListEntry('FinCashList',NULL,'Fin%',NULL);
CALL TDSTATS.AddObjectListEntry('FinCashList',NULL,'Account%', 'Cash%');
CALL TDSTATS.RemoveObjectList('FinCashList',NULL,NumRemoved);
```

## **SelectFromObjectList**

Displays the database and table objects contained within a specified object list.

### **Syntax**

```
REPLACE PROCEDURE TDSTATS.SelectFromObjectList (
  IN ObjectListName   VARCHAR(128) CHARACTER SET UNICODE,
  IN ObjectListId     BIGINT,
  IN ResolveWildCards CHAR(1) CHARACTER SET LATIN
)
...
;
```

### **Syntax Elements**

#### ***ObjectListName***

Name of the existing object list.

If you specify NULL, all object lists in the TDSTATS database are displayed.

### ***ObjectListId***

ID of an existing object list.

### ***ResolveWildCards***

Possible values:

- Y. If you specify Y, object names containing wildcard characters will be expanded.
- N. If you specify N, wildcard characters within object names will be retained. This is the default value.

## **Output**

| Column         | Description                                                                                                                                                                                                             |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ObjectListName | Name of object list containing the object name. For more information, see the ObjectName output parameter.                                                                                                              |
| DatabaseName   | Database whose objects are contained in the object list. For more information, see the ObjectListName output parameter.                                                                                                 |
| ObjectName     | Table contained within the object list. If you specify NULL, all tables in the DatabaseName output parameter are contained in the list. For more information, see the ObjectListName or DatabaseName output parameters. |

## **Usage Notes**

You can view object list data in the TDSTATS.ObjectList and TDSTATS.ObjectListEntry tables. The names of the object lists are stored in the TDSTATS.ObjectList.ListName column.

## **Returning Result Set**

The output of this external stored procedure is in the form of a stored procedure dynamic result. That is, the external stored procedure can return result sets to the client application or to the caller of the external stored procedure (in addition to consuming the result sets itself) upon completion of the external stored procedure.

For more information about stored procedure dynamic result sets, see *Teradata Vantage™ - SQL External Routine Programming*, B035-1147.

## **Displaying Statistics**

| To display ...                           | You must specify ...                                              |
|------------------------------------------|-------------------------------------------------------------------|
| the objects for a specified list         | either a value for <i>ObjectListName</i> or <i>ObjectListId</i> . |
| all lists stored in the TDSTATS database | NULL for <i>ObjectListName</i> and <i>ObjectListId</i> .          |

**Example: Using SelectFromObjectList**

The following example shows how to display all objects in the list named MyObjects.

```
CALL TDSTATS.SelectFromObjectList('MyObjects',NULL,'Y');
*** Query completed. 4 rows found. 1 column returned.
*** Total elapsed time was 1 second.
```

| ObjectListName | DatabaseName | ObjectName |
|----------------|--------------|------------|
| -----          | -----        | -----      |
| MyObjects      | db1          | ?          |
| MyObjects      | db2          | tx         |
| MyObjects      | db3          | tx         |
| MyObjects      | db3          | ty         |
| MyObjects      | db3          | tz         |

## Open APIs for Query Lists

Use these external store procedures to create and maintain query lists.

### AddQueryList

Creates a new query list with a user specified name which can be passed to AnalyzeStatsUsage for analysis.

**Syntax**

```
REPLACE PROCEDURE TDSTATS.AddQueryList (
  IN QueryListName VARCHAR(128) CHARACTER SET UNICODE
)
  ...
;
```

**Syntax Elements*****QueryListName***

Name of the query list.

*QueryListName* must be unique among all query lists stored in the TDSTATS database.

**Usage Notes**

After a new query list is created, you can call the AddQueryListEntry external stored procedure to add queries to a list.

**Example: Using AddQueryList**

The following example shows how to create a list consisting of two queries and analyze statistics usage on them.

**Note:**

The input argument value that begins with a colon represents a host variable whose value was populated in a prior call where it served as an output argument.

```
CALL TDSTATS.AddQueryList('MySlowQueries');
CALL TDSTATS.AddQueryListEntry('MySlowQueries',8675309,1);
CALL TDSTATS.AddQueryListEntry('MySlowQueries',8675325,1);
CALL
TDSTATS.AnalyzeStatsUsage(NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL
,'MySlowQueries',
'N',AnalysisId, NumEvents);
CALL TDSTATS.AnalyzeStatsUsageReport(:AnalysisId);
```

**Related Information**

| For more information on ...                     | See ..                             |
|-------------------------------------------------|------------------------------------|
| the AnalyzeStatsUsage external stored procedure | <a href="#">AnalyzeStatsUsage.</a> |
| adding queries to your list                     | <a href="#">AddQueryListEntry.</a> |

**AddQueryListEntry**

Adds a query to a specified query list where the query is identified by its DBQL ID.

**Syntax**

```
REPLACE PROCEDURE TDSTATS.AddQueryListEntry (
  IN QueryListName VARCHAR(128) CHARACTER SET UNICODE,
  IN QueryId DECIMAL(18,0)),
  IN Frequency INTEGER
)
...
;
```

Syntax Elements

QueryListName

Name of already created query list. For more information, see [AddQueryList](#).

QueryId

DBQL assigned ID for the query as stored in DBC.DBQLogTbl.QueryID.

Frequency

Number of logged instances of this same query.  
If you specify NULL, the assumed frequency is 1.

Usage Notes

Typically, queries in need of performance tuning are added to a query list.  
You can view the query list data in the TDSTATS.QueryList and TDSTATS.QueryListEntry tables and the names of the query lists in the TDSTATS.QueryList.ListName column.

Example: Using AddQueryListEntry

The following example shows how to add queries 8675309 and 8675325 to the query list named MySlowQueries.

Note:

The input parameter value that begins with a colon represents a host variables whose value was populated in a prior call where it served as an output parameter.

```
CALL TDSTATS.AddQueryList('MySlowQueries');
CALL TDSTATS.AddQueryListEntry('MySlowQueries',8675309,1);
CALL TDSTATS.AddQueryListEntry('MySlowQueries',8675325,1);
CALL
TDSTATS.AnalyzeStatsUsage(NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL
,'MySlowQueries',
'N',AnalysisId, NumEvents);
CALL TDSTATS.AnalyzeStatsUsageReport(:AnalysisId);
```

Related Information

| For more information on .... | See ...                        |
|------------------------------|--------------------------------|
| previous created query lists | <a href="#">AddQueryList</a> . |

| For more information on ....     | See ...                                                                                                                                             |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| the DBC.DBQLogTbl.QueryID column | <i>Teradata Vantage™ - Database Administration</i> , B035-1093 or use Teradata Studio or Teradata Studio Express to view details about this column. |

## RemoveQueryList

Deletes a query list previously created by AddQueryList along with all of its entries.

### Syntax

```
REPLACE PROCEDURE TDSTATS.RemoveQueryList (
  IN QueryListName VARCHAR(128) CHARACTER SET UNICODE,
  OUT NumRemoved    INTEGER
)
...
;
```

### Syntax Elements

#### QueryListName

Name of the existing query list created by a call to AddQueryList. For more information, see [AddQueryList](#).

This input parameter cannot be NULL.

#### NumRemoved

Number of query entries removed along with the specified list.

### Usage Notes

To remove only an individual query entry from a list, you must call the RemoveQueryListEntry external stored procedure.

You can view the query list data in the TDSTATS.QueryList and TDSTATS.QueryListEntry tables and the names of the query lists in the TDSTATS.QueryList.ListName column.

### Example: Using RemoveQueryList

The following example shows how to remove the query list named MySlowQueries.

```
CALL TDSTATS.RemoveQueryList('MySlowQueries',NumRemoved);
```

## Related Information

| For more information on ...                    | See ...                               |
|------------------------------------------------|---------------------------------------|
| previous created query lists                   | <a href="#">AddQueryList.</a>         |
| removing an individual query entry from a list | <a href="#">RemoveQueryListEntry.</a> |

## RemoveQueryListEntry

Removes an individual query from a query list.

### Syntax

```
REPLACE PROCEDURE TDSTATS.RemoveQueryListEntr (
  IN QueryListName VARCHAR(128) CHARACTER SET UNICODE,
  IN QueryID       DECIMAL(18,0)
)
...
;
```

### Syntax Elements

#### *QueryListName*

Name of the existing query list containing the query to be removed.

This input parameter cannot be NULL.

#### *QueryID*

DBQL-assigned ID of the query.

This input parameter cannot be NULL.

### Usage Notes

When you specify an individual query to remove, it must have already been added to an existing query list by a call to the AddQueryListEntry external stored procedure. All other queries in the query list will remain intact.

You can view the query list data in the TDSTATS.QueryList and TDSTATS.QueryListEntry tables and the names of the query lists in the TDSTATS.QueryList.ListName column.

### Example: Using RemoveQueryListEntry

The following example shows how to remove an individual query (for example, query 8675309) from the query list named MySlowQueries.



```
CALL TDSTATS.RemoveQueryListEntry('MySlowQueries',8675309);
```

**Related Information**

For more information about the AddQueryListEntry external stored procedure, see [AddQueryListEntry](#).

# Sample PM/API Application

The following is a sample PM/API application based on a sample CLI program. A PM/API application is no more than a CLI application. The only difference between a regular SQL CLI application and a PM/API application is:

- PM/API logs on to a Monitor partition
- SQL logs onto DBC/SQL partition

---

## Note:

This sample PM/API application is for an earlier monitor version. You will need to change the monitor version to return needed PM/API fields used in creating and enhancing applications.

---

For a sample PM/API application that uses the Teradata JDBC Driver, see the sample programs in *Teradata JDBC Driver Reference*, available at <https://teradata-docs.s3.amazonaws.com/doc/connectivity/jdbc/reference/current/frameset.html>.

## Sample Application

The following is a sample of a CLI program.

```

/*****
/* This is a sample PM/API application.                                */
/* 1) Prompts for a logon string                                       */
/* 2) Connects to a Teradata RDBMS and establishes a MONITOR session */
/* 3) Prompts for input parameter for MONITOR SESSION; PM/API command */
/* 4) Submits MONITOR SESSION;                                       */
/* 5) Retrieves the response, formats and prints the response parcels */
/* 6) Above 3 - 5 are repeated until Q is entered in #3.             */
/* 7) Disconnects.  */
*****/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "samplempmc.h"
// Global variables
struct DBCAREA dbc; /* see dbcare.h for definition of structure */
char cnta[4];
int IndicatorMode;
unsigned short TargetVersion = 6;
char RespMode = 'R';

```

```

static int IndicFieldCount = 0;
static char CurrRequestText[32];
#define MON_SESS_REQ 1
// List of field names for Monitor Session Record 2
char *MonSesRec2fieldnamelist[] = {
    "HostId",
    "LogonProcId",
    "RunProcId",
    "SessionNo",
    "UserName",
    "UserAccount",
    "UserId",
    "LSN",
    "LogonTime",
    "LogonDate",
    "PartName",
    "Priority",
    "IFPState",
    "IFPCPUsec",
    "XactCount",
    "ReqCount",
    "ReqCacheHits",
    "AMPState",
    "AMPCPUsec",
    "AMPIO",
    "DeltaAMPSPool",
    "Blk1HostId",
    "Blk1SessNo",
    "Blk1UserId",
    "Blk1LMode",
    "Blk1OType",
    "Blk1ObjDBId",
    "Blk1ObjTId",
    "Blk1Status",
    "Blk2HostId",
    "Blk2SessNo",
    "Blk2UserId",
    "Blk2LMode",
    "Blk2OType",
    "Blk2ObjDBId",
    "Blk2ObjTId",
    "Blk2Status",
    "Blk3HostId",
    "Blk3SessNo",

```

```

    "Blk3UserId",
    "Blk3LMode",
    "Blk3OType",
    "Blk3ObjDBId",
    "Blk3ObjTId",
    "Blk3Status",
    "MoreBlockers",
    "LogonSource",
    "TempSpaceUsage",
    "HotAmp1CPU",
    "HotAmp2CPU",
    "HotAmp3CPU",
    "HotAmp1IO ",
    "HotAmp2IO ",
    "HotAmp3IO ",
    "HotAmp1CPUId",
    "HotAmp2CPUId",
    "HotAmp3CPUId",
    "HotAmp1IOId",
    "HotAmp2IOId",
    "HotAmp3IOId",
    "LowAmp1CPU",
    "LowAmp2CPU",
    "LowAmp3CPU",
    "LowAmp1IO ",
    "LowAmp2IO ",
    "LowAmp3IO ",
    "LowAmp1CPUId",
    "LowAmp2CPUId",
    "LowAmp3CPUId",
    "LowAmp1IOId",
    "LowAmp2IOId",
    "LowAmp3IOId",
    "AmpCount",
    "AvgAmpCPUsec",
    "AvgAmpIOCnt",
    "ReqStartTime",
    "ReqStartDate",
    "ReqCPU",
    "ReqIO",
    "RequestNo",
    "WlcId",
    "DontReclassifyFlag",
};

```

```

/* This routine displays the presence bits along with the field name. */
static void PrintPresenceBits(char *ptr,int fieldcount,char **fieldnamelist)
{
    char *tempPtr=ptr;
    char temp = *ptr;
    int i=0;
    int bitOn;
    char buf[200];
    int endlist=0;
    printf("\nPresence Bits:\n");
    while (i < fieldcount)
    {
        bitOn = (temp & (char)0x80);
        if (!endlist)
        {
            if (!fieldnamelist ||
                (fieldnamelist && strlen(fieldnamelist[i]) == 0))
                endlist = 1;
        };
        if (endlist)
        {
            sprintf(buf,"Field%03d",i+1);
        }
        else
            strcpy(buf,fieldnamelist[i]);
        if (bitOn)
            printf("[%02d] %30s \tis \tNULL\n",i,buf);
        else
            printf("[%02d] %30s \tis \tNOT NULL\n",i,buf);
        i++;
        if ((i % 8) == 0)
        {
            tempPtr++;
            temp = *tempPtr;
        }
        else
            temp = temp << 1;
    }
    printf("\n");
}
/*****
/* WRITE_ERROR -- Prints error message that is located in
/* the response buffer--Error or Failure Parcel */
*****/

```

```

static void Write_error(char *parcel)
{
    int i;
    Error_Fail_t *Error_Fail;
    Error_Fail = ((struct ERROR_FAIL_Type *) (parcel));
    printf("DBS Error code: %d : ", Error_Fail->Code);
    for(i=0;i < (int)Error_Fail->Length;i++)
        printf("%c", Error_Fail->Msg[i]);
    printf("\n");
} /**end write_error**/
/* MONITOR SESSION formatting routines */
static void PrtSes1(char *dataptr)
{
    short *SampleRate;
    SampleRate = (short *) (dataptr);
    printf("Sample rate: %d seconds\n\n", *SampleRate);
}
static void PrtSes2(char *dataptr, int version)
{
    MonSesRec2_t *MonSesRec2;
    MonSesRecV3_t *MonSesRecV3;
    MonSesRecV4_t *MonSesRecV4;
    MonSesRecV5_t *MonSesRecV5;
    MonSesRec2 = (MonSesRec2_t *) (dataptr);
    if (version >= MONVERSION3)
    {
        MonSesRecV3 = (MonSesRecV3_t *) (((char *) (&MonSesRec2->LogonSource)) +
            MonSesRec2->LogonSourceLen);
    }
    if (version >= MONVERSION4)
    {
        MonSesRecV4 = (MonSesRecV4_t *) (MonSesRecV3 +1);
    }
    if (version >= MONVERSION5)
    {
        MonSesRecV5 = (MonSesRecV5_t *) (MonSesRecV4 +1);
    }
    printf("HostId: %8d   ", MonSesRec2->HostId);
    printf("SessionNo: %d\n", MonSesRec2->SessionNo);
    printf("LogonPENo: %5d   ", MonSesRec2->LogonProcId);
    printf("RunVprocNo: %5d\n", MonSesRec2->RunProcId);
    printf("PartName: %.16s", MonSesRec2->PartName);
    printf("PEState:  %.18s\n\n", MonSesRec2->PEState);
    printf("LogonDate: %04d/%02d/%02d      ",

```

```

        (int) MonSesRec2->LogonDate / 10000 + 1900,
        (int) MonSesRec2->LogonDate / 100 % 100,
        (int) MonSesRec2->LogonDate % 100);
printf("LogonTime: %02d:%02d:%05.2f\n",
        (int) MonSesRec2->LogonTime / 10000,
        (int) MonSesRec2->LogonTime / 100 % 100,
        (int) MonSesRec2->LogonTime % 100 +
        MonSesRec2->LogonTime -
        (int) MonSesRec2->LogonTime);
printf("UserID: %d ", MonSesRec2->UserID);
printf("LSN: %d\n", MonSesRec2->LSN);
printf("UserName: %.30s\n", MonSesRec2->UserName);
printf("UserAccount: %.30s\n\n", MonSesRec2->UserAccount);
printf("PECPUSec: %10.2f ", MonSesRec2->PECPUSec);
printf("XactCount: %13.2f\n", MonSesRec2->XactCount);
printf("ReqCount: %10.1f ", MonSesRec2->ReqCount);
printf("ReqCacheHits: %10.1f\n\n", MonSesRec2->ReqCacheHits);
printf("AMPState: %.18s\n", MonSesRec2->AMPState);
printf("AMPCPUSec: %9.2f ", MonSesRec2->AMPCPUSec);
printf("AMPIO: %7.2f\n", MonSesRec2->AMPIO);
printf("Request_AMPSPool: %4.1f\n\n", MonSesRec2->Delta_AMPSPool);
printf("Blk_1_HostId: %6d ", MonSesRec2->Blk_1_HostId);
printf("Blk_2_HostId: %6d ", MonSesRec2->Blk_2_HostId);
printf("Blk_3_HostId: %6d\n", MonSesRec2->Blk_3_HostId);
printf("Blk_1_SessNo: %6d ", MonSesRec2->Blk_1_SessNo);
printf("Blk_2_SessNo: %6d ", MonSesRec2->Blk_2_SessNo);
printf("Blk_3_SessNo: %6d\n", MonSesRec2->Blk_3_SessNo);
printf("Blk_1_UserID: %6d ", MonSesRec2->Blk_1_UserID);
printf("Blk_2_UserID: %6d ", MonSesRec2->Blk_2_UserID);
printf("Blk_3_UserID: %6d\n", MonSesRec2->Blk_3_UserID);
printf("Blk_1_LMode: %c ", MonSesRec2->Blk_1_LMode);
printf("Blk_2_LMode: %c ", MonSesRec2->Blk_2_LMode);
printf("Blk_3_LMode: %c\n", MonSesRec2->Blk_3_LMode);
printf("Blk_1_OType: %c ", MonSesRec2->Blk_1_OType);
printf("Blk_2_OType: %c ", MonSesRec2->Blk_2_OType);
printf("Blk_3_OType: %c\n", MonSesRec2->Blk_3_OType);
printf("Blk_1_ObjDBId: %5d ", MonSesRec2->Blk_1_ObjDBId);
printf("Blk_2_ObjDBId: %5d ", MonSesRec2->Blk_2_ObjDBId);
printf("Blk_3_ObjDBId: %5d\n", MonSesRec2->Blk_3_ObjDBId);
printf("Blk_1_ObjTId: %6d ", MonSesRec2->Blk_1_ObjTId);
printf("Blk_2_ObjTId: %6d ", MonSesRec2->Blk_2_ObjTId);
printf("Blk_3_ObjTId: %6d\n", MonSesRec2->Blk_3_ObjTId);

printf("Blk_1_Status: %c ", MonSesRec2->Blk_1_Status);

```

```

printf("Blk_2_Status: %c      ", MonSesRec2->Blk_2_Status);
printf("Blk_3_Status: %c\n", MonSesRec2->Blk_3_Status);
printf("MoreBlockers: %c\n\n", MonSesRec2->MoreBlockers);
MonSesRec2->LogonSource[MonSesRec2->LogonSourceLen] = '\0';
printf("LogonSource: %.128s\n\n", MonSesRec2->LogonSource);
/* MonVerId 3 fields */
if (version < MONVERSION3)
    return;
printf("HotAmp1CPU: %9.2f      ", MonSesRecV3->HotAmp1CPU);
printf("HotAmp2CPU: %9.2f      ", MonSesRecV3->HotAmp2CPU);
printf("HotAmp3CPU: %9.2f\n", MonSesRecV3->HotAmp3CPU);
printf("HotAmp1CPUId: %7d      ", MonSesRecV3->HotAmp1CPUId);
printf("HotAmp2CPUId: %7d      ", MonSesRecV3->HotAmp2CPUId);
printf("HotAmp3CPUId: %7d\n\n", MonSesRecV3->HotAmp3CPUId);
printf("HotAmp1IO: %10.2f      ", MonSesRecV3->HotAmp1IO);
printf("HotAmp2IO: %10.2f      ", MonSesRecV3->HotAmp2IO);
printf("HotAmp3IO: %10.2f\n", MonSesRecV3->HotAmp3IO);
printf("HotAmp1IOId: %8d      ", MonSesRecV3->HotAmp1IOId);
printf("HotAmp2IOId: %8d      ", MonSesRecV3->HotAmp2IOId);
printf("HotAmp3IOId: %8d\n\n", MonSesRecV3->HotAmp3IOId);
printf("LowAmp1CPU: %9.2f      ", MonSesRecV3->LowAmp1CPU);
printf("LowAmp2CPU: %9.2f      ", MonSesRecV3->LowAmp2CPU);
printf("LowAmp3CPU: %9.2f\n", MonSesRecV3->LowAmp3CPU);
printf("LowAmp1CPUId: %7d      ", MonSesRecV3->LowAmp1CPUId);
printf("LowAmp2CPUId: %7d      ", MonSesRecV3->LowAmp2CPUId);
printf("LowAmp3CPUId: %7d\n\n", MonSesRecV3->LowAmp3CPUId);
printf("LowAmp1IO: %10.2f      ", MonSesRecV3->LowAmp1IO);
printf("LowAmp2IO: %10.2f      ", MonSesRecV3->LowAmp2IO);
printf("LowAmp3IO: %10.2f\n", MonSesRecV3->LowAmp3IO);
printf("LowAmp1IOId: %8d      ", MonSesRecV3->LowAmp1IOId);
printf("LowAmp2IOId: %8d      ", MonSesRecV3->LowAmp2IOId);
printf("LowAmp3IOId: %8d\n\n", MonSesRecV3->LowAmp3IOId);
printf("AvgAmpCPUsec:%8.2f      ", MonSesRecV3->AvgAmpCPUsec);
printf("AvgAmpIOCnt: %8.2f\n", MonSesRecV3->AvgAmpIOCnt);
printf("AmpCount: %11d\n\n", MonSesRecV3->AmpCount);
printf("TempSpaceUsg: %7.2f\n\n", MonSesRecV3->TempSpaceUsage);
if (version < MONVERSION4)
    return;
/* MonVerId 4 fields */
printf("ReqStartTime: %04d/%02d/%02d ",
      (int) MonSesRecV4->ReqStartDate / 10000 + 1900,
      (int) MonSesRecV4->ReqStartDate / 100 % 100,
      (int) MonSesRecV4->ReqStartDate % 100);
printf("%02d:%02d:%05.2f",

```



```

        (int) MonSesRecV4->ReqStartTime / 10000,
        (int) MonSesRecV4->ReqStartTime / 100 % 100,
        (int) MonSesRecV4->ReqStartTime % 100 +
        MonSesRecV4->ReqStartTime -
        (int) MonSesRecV4->ReqStartTime);
printf("  ReqCPU: %10.2f", MonSesRecV4->ReqCPU);
printf("  ReqIO: %10.2f\n", MonSesRecV4->ReqIO);
if (version < MONVERSION5)
    return;
/* MonVerId 5 fields */
printf("ReqNo:%d   WlcId: %d       DontReclassifyFlag: %d\n",
        MonSesRecV5->ReqNo,
        MonSesRecV5->WlcId, MonSesRecV5->DontReclassifyFlag);
}
static void CheckSuccess(int ActivityType,
                        int ActivityCount,
                        int StatementNo)
{
    switch (ActivityType)
    {
        case PCLSTMTNULL:
            printf("Null statement successful. \n\n");
            break;
    }
    return;
}
static void CheckData(int ActivityType, int StatementNo)
{
    DataInfo_t * DataInfo;
    int i;
    DataInfo = (DataInfo_t *) (dbc.fet_data_ptr);
    printf("\nCheckData called for Indicator mode data info parcel.\n");
    printf("Field Count: %d \n", DataInfo->field_count);
    IndicFieldCount = DataInfo->field_count;

    if (DataInfo->field_count > MAXFIELDS)
        DataInfo->field_count = MAXFIELDS;
    for (i = 0; i < DataInfo->field_count; i++)
    {
        printf("Field %d: Type = %d Size = %d\n", i + 1,
            DataInfo->FieldData[i].FType,
            DataInfo->FieldData[i].FSize);
    }
    return;
}

```

```

}
static void CheckRecord(int ActivityType, int StatementNo, int ActivityCount)
{
    switch (ActivityType)
    {
        case PCLMONSESS:
            if (StatementNo == 1)
                PrtSes1(dbc.fet_data_ptr);
            else if (StatementNo == 2)
            {
                int IndLen=0;
                if (IndicatorMode)
                {
                    switch (TargetVersion)
                    {
                        case MONVERSION3:
                        case MONVERSION4:
                            IndLen = 10;
                            break;
                        case MONVERSION5:
                        case MONVERSION6:
                        default:
                            IndLen = 11;
                            break;
                    }
                    PrintPresenceBits(dbc.fet_data_ptr,IndicFieldCount,MonSesRec2fieldnamelist);
                    dbc.fet_data_ptr = (char *)dbc.fet_data_ptr + IndLen;
                }
                PrtSes2(dbc.fet_data_ptr,TargetVersion);
                printf("\n");
                printf("-----\n\n");
            }
            break;
        default:
            printf("\nFound a funny parcel number: %d\n",ActivityType);
            break;
    }
}

/*****
/* SET_OPTIONS -- sets cli options (record,field,indicator
/* mode; etc..)
/* Sets size of buffers (request and response) */
*****/
set_options()

```

```

{
    dbc.change_opts = 'Y';
    dbc.resp_mode = 'R';
    dbc.use_presence_bits = 'Y';
    dbc.keep_resp = 'N';
    dbc.wait_across_crash = 'N';
    dbc.tell_about_crash = 'Y';
    dbc.loc_mode = 'Y';
    dbc.var_len_req = 'N';
    dbc.var_len_fetch = 'N';
    dbc.save_resp_buf = 'N';
    dbc.two_resp_bufs = 'N';
    dbc.ret_time = 'N';
    dbc.parcel_mode = 'Y';
    dbc.wait_for_resp = 'Y';
    dbc.req_proc_opt = 'E';
    dbc.req_buf_len = 256;
    dbc.resp_buf_len = 32000;
    return(0);
} /**end set_options**/
/*****
/* EXITOUT -- If connected logs off session. Cleans up          */
/*          any used memory.                                     */
*****/
void exitout(int status)
{
    Int32 result;
    if (status == CONNECTED)
    {
        printf ("Logging off.\n");
        dbc.func = DBFDSC;
        DBCHCL (&result,cnta,&dbc);
        if (result != EM_OK)
            printf("disconnect failed -- %s\n", dbc.msg_text);
    }
    DBCHCLN(&result,cnta);
    if (result != EM_OK)
        printf("cleanup failed -- %s\n", dbc.msg_text);
    exit(0);
} /**end exitout**/
/*****
/* INITIALIZE_DBCAREA -- Sets default options (clispb.dat) in   */
/*          dbcarea by calling DBCHINI()                        */
*****/

```

```

initialize_dbcarea()
{
    Int32  result;
    dbc.total_len = sizeof(struct DBCAREA);
    DBCHINI(&result,cnta,&dbc);
    if (result != EM_OK)
    { /* if we can't initialize, we can't go on, so exit */
        printf("init failed -- %s\n", dbc.msg_text);
        exitout(NOT_CONNECTED);
    }
    return(0);
} /*end initialize_dbcarea*/
/*****
/* LOGON_TO_DBC -- Connects session and logs on to DBC          */
/*                  Fetches result of connection to check if      */
/*                  successful                                     */
*****/
logon_to_dbc(char * logonstr)
{
    struct CliCONNECTType connectstr;
    Int32  result, i;
    char    TempString[30];
    for (i = 0; i < 30; i++)
        TempString[i] = ' ';
    for (i = 0; i < 30; i++)
    {
        if (logonstr[i] == '/')
        {
            TempString[i] = '\0';
            break;
        }
        TempString[i] = logonstr[i];
    }
    printf("Logging on to %s ...\n", TempString);
    dbc.logon_ptr = logonstr;
    dbc.logon_len = strlen(logonstr);
    dbc.func = DBFCON;
    dbc.run_ptr = (char *) &connectstr;
    strncpy(connectstr.PartitionName, "MONITOR", 16);
    connectstr.Function = 0;
    dbc.run_len = sizeof(struct CliCONNECTType);
    DBCHCL(&result,cnta,&dbc);
    if (result != EM_OK) /*if connect fails exit*/
    {

```

```

        printf("connect failed -- %s\n",dbc.msg_text);
        exitout(NOT_CONNECTED);
    }
    printf("Logon successful.\n");
    return(0);
} /**end logon_to_dbc**/
/*****
/* CLOSE_REQUEST -- Terminates and cleans up specified request */
*****/
close_request(Int32 req_id, Int32 sess_id)
{
    Int32 result;
    dbc.i_sess_id = sess_id;
    dbc.i_req_id = req_id;
    dbc.func=DBFERQ;
    DBCHCL(&result,cnta,&dbc);
    if (result !=EM_OK)
    {
        printf("end request failed -- %s\n", dbc.msg_text);
        exitout(CONNECTED);
    }
    return(0);
} /**end close_request**/
/*****
/* OPEN_REQUEST -- Sends request to DBC */
*****/
open_request (int InReqType,
              void *UsingPtr)
{
    Int32 result;
    char CurrRequestText[32];

    /* set up for a new request */
    dbc.func = DBFIRQ;
    dbc.using_data_ptr = (char *) UsingPtr;

    switch (InReqType)
    {
        case MON_SESS_REQ:
            strcpy(CurrRequestText,"MONITOR SESSION;");
            dbc.using_data_len = sizeof(monssess_t);
            break;
        default:
            printf("Internal error\n");
    }
}

```

```

        exitout(CONNECTED);
    }
    dbc.req_ptr = &CurrRequestText[0];
    dbc.req_len = strlen (CurrRequestText);
    printf ("\nSubmitting request %s ...\n", dbc.req_ptr);

    /* Submit request */
    DBCHCL (&result,cnta,&dbc);
    if (result != EM_OK)
    {
        printf ("result = %d \n", result);
        printf ("initiate request failed -- %s \n", dbc.msg_text);
        exitout(CONNECTED);
    }
    return(EM_OK);
} /**end open_request**/
int fetch_request(Int32 request, Int32 session)
{
    int      status;
    int      StatementNo = 0;
    int      ActivityType = 0;
    int      ActivityCount = 0;
    Int32    result;
    int      ReturnCode;
    dbc.i_sess_id = session;
    dbc.i_req_id = request;
    dbc.func = DBFFET;
    status = OK;
    ReturnCode = OK;
    /* fetch one parcel at a time until all parcels are used up */
    /* or an error occurs */
    while (status == OK)
    {
        DBCHCL(&result,cnta,&dbc);
        if (result == REQEXHAUST)
            status = STOP;
        else if (result != EM_OK)
            status = NOT_OK;
        else
        {
            switch ((Int16) (dbc.fet_parcel_flavor))
            {
                case PCLSUCCESS :
                    /*Success Parcel */
            }
        }
    }
}

```

```

        struct PclSUCCESSType succ;
        memcpy(&succ, dbc.fet_data_ptr-4, 4);
        memcpy(&succ, dbc.fet_data_ptr-4, succ.Length+4);
        StatementNo = succ.StatementNo;
        ActivityCount = *(int *)&succ.ActivityCount[0];
        ActivityType = succ.ActivityType;
        if ( succ.WarningLength > 0 )
            printf("%s \n", succ.WarningMsg);
        CheckSuccess(ActivityType, ActivityCount, StatementNo);
    }
    break;
case PclDATAINFO :
    CheckData(ActivityType, StatementNo);
    break;
case PclRECORD :
    /*Returned data */
    CheckRecord(ActivityType, StatementNo, ActivityCount);
    break;
case PclFAILURE :
    /*Failure parcel*/
case PclERROR :
    /*Error parcel */
    status = STOP;
    Write_error(dbc.fet_data_ptr);
    ReturnCode = PclERROR;
    break;
} /*switch*/
}
} /*while*/
if (status == NOT_OK)
{
    printf("fetch failed -- %s \n", dbc.msg_text);
    exitout(CONNECTED);
} /*if*/
return(ReturnCode);
} /**end fetch_request*/
static Boolean MonSession()
{
    monsess_t MonSess;
    int i;
    char InString[MAXNAME];
    /* Initializations */
    MonSess.monheader.version = TargetVersion;
    MonSess.monheader.indicbyte = '\0';
    for (i = 0; i < MAXNAME; i++)
        monsess.user[i] = ' ';
    printf("Enter logical hostid (* for all hosts) or Q)uit: ");

```

```

gets(InString);
if (!strcmp(InString, "*") || (strlen(InString) == 0))
    MonSess.hostid = (Int16) 65535;
else if (!strcmp(InString, "Q") || !strcmp(InString, "q"))
    return (TRUE);
else
    MonSess.hostid = atoi(InString);
printf("Enter user name (* for all users): ");
gets(InString);
if (!strcmp(InString, "*") || (strlen(InString) == 0))
{
    for (i = 0; i < MAXNAME; i++)
    {
        if (InString[i] == '\0')
            break;
        MonSess.user[i] = InString[i];
    }
}
printf("Enter session number (* for all sessions): ");
gets(InString);
if (!strcmp(InString, "*") || (strlen(InString) == 0))
    MonSess.session = 0;
else
    MonSess.session = atoi(InString);
if (RespMode == 'I' || RespMode == 'i')
{
    dbc.resp_mode = 'I';
    IndicatorMode = TRUE;
}
else
{
    dbc.resp_mode = 'R';
    IndicatorMode = FALSE;
}
open_request( MON_SESS_REQ,&MonSess);
fetch_request(dbc.o_req_id, dbc.o_sess_id);
close_request(dbc.o_req_id, dbc.o_sess_id);
return (FALSE);
}
main (int argc, char **argv)
{
    char    LogonString[MAXLOGONSIZE];
    Boolean stop = FALSE;
    initialize_dbcarea();

```



```

set_options();

//6: Teradata 12.x; 5: Teradata 6.x 4: Teradata 5.x
TargetVersion = 6;
// 'R' no presence bits returned; 'I' presence bits returned
RespMode = 'I';
IndicatorMode = TRUE;
printf ("TargetVersion(argv1) = %d, RespMode(argv2) = %c \n\n",
        TargetVersion, RespMode);
/* Establish session */
printf ("Enter logon string (tdpid/user,password): ");
gets (LogonString);
if (strlen(LogonString) == 0)
{
    printf ("NULL logon string not accepted.\n");
    return (1);
}

logon_to_dbc(LogonString);
if (fetch_request(dbc.o_req_id,dbc.o_sess_id) != OK)
    exitout(NOT_CONNECTED);
close_request(dbc.o_req_id,dbc.o_sess_id);
while (stop == FALSE)
    stop = MonSession();
/* Logoff and cleanup */
exitout(CONNECTED);
return 0;
} /**end main**/

```

## Header File for Sample PMPC Application

This section defines the contents for all parcel flavors. Each parcel is defined in the Notes section under the definition of each parcel.

```

#ifndef PMPC_H
#define PMPC_H
/* These are standard CLI include files */
#include "coptypes.h"
#include "coperr.h"
#include "dbcarea.h"
#include "parcel.h"
/*****
/* Purpose To define the data type for kinds of activities

```

```

        returned in the Ok and Success parcels. */
typedef unsigned short pclstmt_t;
/* Purpose To express the kind of activity within Ok and Success parcels. */
#define PCLSTMTNUL          (0)
#define PCLRETSTMT         (1)
#define PCLINSSTMT         (2)
#define PCLUPDSTMT          (3) /* UPDATE                      */
#define PCLUPDRETSTMT       (4) /* UPDATE ... RETRIEVING */
#define PCLDELSTMT         (5)
#define PCLCTSTMT          (6)
#define PCLMODTABSTMT       (7)
#define PCLCVSTMT          (8)
#define PCLCMSTMT          (9)
#define PCLDROPTABSTMT      (10)
#define PCLDROPVIEWSTMT     (11)
#define PCLDROPMACSTMT      (12)
#define PCLDROPINDSTMT      (13)
#define PCLRENTABSTMTT      (14)
#define PCLRENVIEWSTMT      (15)
#define PCLRENMACSTMT       (16)
#define PCLCREINDSTMT       (17)
#define PCLCDSTMTT          (18)
#define PCLCREUSERSTMT      (19)
#define PCLGRANTSTMT        (20)
#define PCLREVOKESTMT       (21)
#define PCLGIVESTMT         (22)
#define PCLDROPDBSTMT       (23)
#define PCLMODDBSTMT        (24)
#define PCLDATABASESTMT     (25)
#define PCLBTSTMT           (26)
#define PCLETSTMT           (27)
#define PCLABORTSTMT        (28)
#define PCLNULLSTMT         (29)
#define PCLEXECSTMT         (30)
#define PCLCMNTSETSTMT      (31) /* COMMENT set statement */
#define PCLCMNTGETSTMT      (32) /* COMMENT returning statement */
#define PCLEHOSTMT          (33)
#define PCLREPVIEWSTMT      (34)
#define PCLREPMACSTMT       (35)
#define PCLCHECKPTSTMT      (36)
#define PCLDELJRNLTSTMT     (37)
#define PCLROLLBACKSTMT     (38)
#define PCLRELLOCKSTMT      (39)
#define PCLHUTCONFIGSTMT    (40)

```

```

#define PCLVCHECKPTSTMT      (41)
#define PCLDUMPJRNLSMT      (42)
#define PCLDUMPDBSTMT       (43)
#define PCLRESTORESTMT      (44)
#define PCLROLLFORWSTMT     (45)
#define PCLDELDDBSTMT       (46)
#define PCLCLEARDUMPST      (47)
#define PCLSAVEDUMPSTMT     (48)
#define PCLSHOWSTMT         (49)
#define PCLHELPSTMT         (50)
#define PCLBEGINLOADSTMT    (51)
#define PCLCHKPTLOADSTMT    (52)
#define PCLENDLOADSTMT      (53)
#define PCLLINSTMT          (54)
#define PCLGRANTLOGONSTMT   (55)
#define PCLREVOKELOGONSTMT (56)
#define PCLBEGACCTLOGSTMT   (57)
#define PCLENDACCTLOGSTMT   (58)
#define PCLCOLLSTATSTMT     (59)
#define PCLDROPSTATSTMT     (60)
#define PCLSESSESTMT        (61)
#define PCLBEGEDITSTMT      (62)
#define PCLEEDITSTMT        (63)
#define PCLEXECEDITSTM      (64)
#define PCLENDEDITSTMT      (65)
#define PCLREEDITSTMT      (66)
#define PCLEEDITDELSTMT     (67)
#define PCLEEDITINSSTMT     (68)
#define PCLEEDITUPDSTMT     (69)
#define PCLBEGDEMLSTMT      (70)
#define PCLDATASTATUS       (71)
#define PCLBEGEXPORTSTMT    (74)
#define PCLENDEXPORTSTMT    (75)
#define PCL2PCVOTEREQ       (76) /* 2PC vote request.          */
#define PCL2PCVOTETERM      (77) /* 2PC vote and terminate.     */
#define PCL2PCCMMT          (78) /* 2PC commit request.        */
#define PCL2PCABRT          (79) /* 2PC abort request.          */
#define PCL2PCFORGET         (80) /* Vote request yes/done       */
#define PCLSETSESSR          (83) /* Set Session Rate            */
#define PCLMONSESS           (84) /* Monitor Session             */
#define PCLIDENTIFY          (85) /* Identify                    */
#define PCLABTSESS           (86) /* Abort Session               */
#define PCLSETRESSR          (87) /* Set Resource Rate           */
#define PCLREVALIDATERISTMT  (89)

```

```

#define PCLCOMMITSTMT      (90)
#define PCLMONVCONFIG      (91) /* Monitor Virtual Config */
#define PCLMONPCONFIG      (92) /* Monitor Physical Config */
#define PCLMONVSUMMARY     (93) /* Monitor Virtual Summary */
#define PCLMONPSUMMARY     (94) /* Monitor Physical Summary */
#define PCLMONVRES         (95) /* Monitor Virtual Resource */
#define PCLMONPRES         (96) /* Monitor Physical Resource */
#define PCLCTRIGSTMT       (97) /* Create Trigger Statement */
#define PCLDTRIGSTMT       (98) /* Drop Trigger Statement */
#define PCLRENTTRIGSTMT    (99) /* Rename Trigger Statement */
#define PCLREPTRIGSTMT     (100) /* Rename Trigger Statement */
#define PCLALTTRIGSTMT     (101) /* Alter Trigger Statement */
#define PCLRDLSTMT         (102) /* Replication statement */
#define PCLDROPPROCSTMT    (103) /* DROP PROCEDURE SQL */
#define PCLCREATEPROCSTMT  (104) /* CREATE PROCEDURE SQL */
#define PCLCALLSTMT        (105) /* CALL SQL */
#define PCLRENPROCSTMT     (106) /* RENAME PROCEDURE SQL */
#define PCLREPPROCSTMT     (107) /* REPLACE PROCEDURE SQL */
#define PCLSETACCT         (108) /* Set Session Account */
#define PCLHULSTMT         (109) /* Arcmain Lock Request */
#define PCLMONSQL          (110) /* MONITOR SQL */
#define PCLMONVER          (111) /* Monitor Version */
#define PCLBEGINDBQLSTMT   (112) /* Begin DBQL */
#define PCLENDDBQLSTMT     (113) /* End DBQL */
#define PCLCREROLESTMT     (114) /* Create Role SQL */
#define PCLDRPROLESTMT     (115) /* Drop Role SQL */
#define PCLGRANTROLESTMT   (116) /* Grant Role SQL */
#define PCLREVOKEROLESTMT  (117) /* Revoke Role SQL */
#define PCLCREPROFILESTMT  (118) /* Create Profile SQL */
#define PCLMODPROFILESTMT  (119) /* Modify Profile SQL */
#define PCLDRPPROFILESTMT  (120) /* Drop Profile SQL */
#define PCLSETROLESTMT     (121) /* Set Role SQL */
#define PCLCREUDFSTMT      (122) /* Create UDF stmt */
#define PCLRPLCUDFSTMT     (123) /* Replace UDF stmt */
#define PCLDROPUDFSTMT     (124) /* Drop UDF stmt */
#define PCLALTERUDFSTMT    (125) /* Alter UDF stmt */
#define PCLRENUDFSTMT      (126) /* Rename UDF stmt */
#define PCLMRGMIXEDSTMT    (127) /* Mixture of upd & ins */
#define PCLMRGUPDSTMT      (128) /* Upds and no ins */
#define PCLMRGINSSTMT      (129) /* Ins and no upds */
#define PCLALTERPROCSTMT   (130) /* ALTER PROCEDURE SQL */
#define PCLTWMSTATSSTMT    (132) /* TDQM Statistics */
#define PCLTWMPERFGROUPSSTMT (133) /* TDQM get Perf Groups */
#define PCLCREUDTSTMT      (134) /* Create UDT stmt */

```

```

#define PCLDROPUDTSTMT      (135) /* Drop UDT stmt */
#define PCLALTERUDTSTMT     (136) /* Alter UDT stmt */
#define PCLRPLCUDTSTMT      (137) /* Replace UDT stmt */
#define PCLCREUDMSTMT       (138) /* Create UDM stmt */
#define PCLALTERUDMSTMT     (139) /* Alter UDM stmt */
#define PCLRPLCUDMSTMT      (140) /* Replace UDM stmt */
#define PCLCRECASTSTMT      (141) /* Create Cast stmt */
#define PCLRPLCCASTSTMT     (142) /* Replace Cast stmt */
#define PCLDROPCASTSTMT     (143) /* Drop Cast stmt */
#define PCLCREORDSTMT       (144) /* Create Ordering stmt */
#define PCLRPLCORDSTMT      (145) /* Replace Ordering stmt */
#define PCLDROPORDSTMT      (146) /* Drop Ordering stmt */
#define PCLCREXFORMSTMT     (147) /* Create Transform stmt */
#define PCLRPLCXFORMSTMT    (148) /* Replace Transform stmt */
#define PCLDROPXFORMSTMT    (149) /* Drop Transform stmt */
#define PCLCREAUTHSTMT      (150) /* Create Auth stmt */
#define PCLDRPAUTHSTMT      (151) /* Drop Auth Stmt */
#define PCLTWMDELREQSTCHGSTM (155) /* TDQM Delay Request Change */
#define PCLTWMSUMMARYSTMT   (156) /* TDQM get Summary */
#define PCLTWMDYNBUILDSTMT  (160) /* TDWM Dynamic Build */
#define PCLTWMLISTWDSTMT    (161) /* TDWM LIST WD */
#define PCLSETSEISOLVLSTMT  (162) /* Set Session Isolation Level */
#define PCLINITIDXANALYSIS  (163) /* Initiate Index Analysis */
#define PCLRPLCAUTHSTMT     (164) /* Replace Auth stmt */
#define PCLSETQBANDSTMT     (165) /* Set QUERY_BAND stmt */
#define PCLLOGARCONSTMT     (166) /* LOGGING ONLINE ARCHIVE ON */
#define PCLLOGARCOFFSTMT    (167) /* LOGGING ONLINE ARCHIVE OFF */
#define PCLMONQUERYBANDSTMT (168) /* MONITOR QUERYBAND */
#define PCLLOGARCONSTMT     (166) /* LOGGING ONLINE ARCHIVE ON */
#define PCLLOGARCOFFSTMT    (167) /* LOGGING ONLINE ARCHIVE OFF */
#define PCLCRECORRSTMT      (169)
#define PCLREPCORRSTMT      (170)
#define PCLDRPCORRSTMT      (171)
#define PCLALTCORRSTMT      (172)
#define PCLUSEREVENTCONTROLSTMT (173) /* USER EVENT CONTROL */
#define PCLEVENTSTATUSSTMT   (174) /* EVENT STATUS */
#define PCLMONAWTRES         (175) /* MONITOR AWT RESOURCE */
#define PCLSPDYNRESULTSET    (176) /* SP Dynamic Result Set */
#define MONVERSION3          3
#define MONVERSION4          4
#define MONVERSION           MONVERSION4
#define CONNECTED            0
#define NOT_CONNECTED        1
#define OK                    0

```

```

#define STOP          1
#define NOT_OK        -1
#define MAXLOGONSIZE 100
#define MAXCOMMANDSIZE 50
#define MAXNAME 30
#define SYSMAXRECSIZE 64000
#define ABORTSESTXT "ABORT SESSION;"
#define SETSESSACCTTXT "SET SESSION ACCOUNT;"
#define MONSESTXT "MONITOR SESSION;"
#define MONSQLTXT "MONITOR SQL;"
#define TDWMPERFGROUPSTXT "TDWM PERFGROUPS;"
#define TDWMSTATISTICSTXT "TDWM STATISTICS;"
#define MONVERTXT "MONITOR VERSION;"
#define SETSESSTXT "SET SESSION RATE;"
#define SETRESTXT "SET RESOURCE RATE;"
#define MONPHYSUMTXT "MONITOR PHYSICAL SUMMARY;"
#define MONVIRSUMTXT "MONITOR VIRTUAL SUMMARY;"
#define MONVIRRESTXT "MONITOR VIRTUAL RESOURCE;"
#define MONPHYRESTXT "MONITOR PHYSICAL RESOURCE;"
#define MONVIRCONTXT "MONITOR VIRTUAL CONFIG;"
#define MONPHYCONTXT "MONITOR PHYSICAL CONFIG;"
#define IDENTSESTXT "IDENTIFY SESSION;"
#define IDENTDBTXT "IDENTIFY DATABASE;"
#define IDENTTABLETXT "IDENTIFY TABLE;"
#define IDENTUSERTXT "IDENTIFY USER;"
#define HELPTXT "HELP;"
typedef short SysInt32[2];
typedef double fltreal64_t;
typedef unsigned short tospocnum_t;
typedef short Integer;
typedef enum Request_t
{
    ABORTSESSREQ,
    SETSESSACCTREQ,
    MONSESSREQ,
    MONSQLREQ,
    TDWMPERFGROUPSREQ,
    TDWMSTATISTICSREQ,
    MONVERREQ,
    MONPHYSUM,
    MONVIRSUM,
    MONVIRRES,
    MONPHYRES,
    MONVIRCON,

```

```

    MONPHYCON,
    SETSESSREQ,
    SETRESREQ,
    IDENTSESREQ,
    IDENTDBREQ,
    IDENTTABLEREQ,
    IDENTUSERREQ
} Request_t;
#pragma pack(1)
/* Input parcel structure for PMPC requests */
typedef struct
{
    char  indicbyte;
    Int16 version;
} monheader_t;
typedef struct
{
    monheader_t monheader;
    Int16 hostid;
    Int32 session;
    char  user[30];
    char  listsess;
    char  logoffsess;
    char  override;
} abortsess_t;
typedef struct
{
    monheader_t monheader;
    Int16 hostid;
    Int32 session;
    char  acct[30];
    char  entiresess;
} setsessacct_t;
typedef struct
{
    monheader_t monheader;
    Int16 hostid;
    Int32 session;
    char  user[30];
} monsess_t;
typedef struct
{
    monheader_t monheader;
    Int16 hostid;

```

```

    Int32 session;
} monsql_t;
typedef struct
{
    monheader_t monheader;
    Int16 samplerate;
    char LocalChange;
    char VirtualChange;
} setrate_t;
typedef struct
{
    Int16 hostid;
    Int32 sessionno;
} identses_t;
typedef struct
{
    monheader_t monheader;
    union
    {
        Int32 objid;
        identses_t identses;
    } UU;
} identify_t;
/* Output record parcel structures */
typedef struct
{
    Int16 HostId; /* Logical Host Identifier */
    char UserName[30]; /* User (database) Name of this session */
    SysInt32 SessionNo; /* Session Number of session */
    char AbortStatus; /* Status of aborted session */
} MonAbtSes_t;
typedef struct
{
    Int16 HostId; /* Logical Host Identifier */
    Int16 LogonProcId; /* Processor number session logged on to */
    Int16 RunProcId; /* Procnum the session initiates requests to */
    Int32 SessionNo; /* Session Number of session */
    char UserName[30]; /* User (database) Name of this session */
    char UserAccount[30]; /* Acct name that this User logged in with */
    Int32 UserID; /* User (database) Identifier for this session */
    Int32 LSN; /* Logon Sequence Number */
    double LogonTime; /* Logon Time for this session */
    Int32 LogonDate; /* Logon Date */
    char PartName[16]; /* Session Partition Name this session is associated with */

```



```

char Priority[2]; /* DBC run priority (Low, Medium, High, Rush) */
char PESTate[18]; /* Current state of this session in the PE */
double PECPUsec; /* Current elapsed cpu usage in PE for sess */
double XactCount; /* Number of transactions processed by a DBC SQL session */
double ReqCount; /* Number of requests processed by a session */
double ReqCacheHits; /* Number of Request Cache uses inDBC/SQL sess */
char AMPState[18]; /* Current state of associated session in AMP processors */
double AMPCPUsec; /* Current elapsed cpu usage in AMP ( TOTAL ) */
double AMPIO; /* Current Total Logical I/O count for associated session */
double Delta_AMPSpool; /* Total Spool space change for the associated session */
Int16 Blk_1_HostId; /* Logical HostId of a session causing block */
Int32 Blk_1_SessNo; /* Session Number of a session causing block */
Int32 Blk_1_UserID; /* User Identifier of a Host Utility Job causing block */
char Blk_1_LMode; /* Mode (Severity) of lock involved in causing a block */
char Blk_1_OType; /* Type (Database/Table/Row Hash) of object being locked */
Int32 Blk_1_ObjDBId; /* DB Identifier of object being locked */
Int32 Blk_1_ObjTId; /* TBID (If applicable) of object being locked */
char Blk_1_Status;
Int16 Blk_2_HostId;
Int32 Blk_2_SessNo;
Int32 Blk_2_UserID;
char Blk_2_LMode;
char Blk_2_OType;
Int32 Blk_2_ObjDBId;
Int32 Blk_2_ObjTId;
char Blk_2_Status;
Int16 Blk_3_HostId;
Int32 Blk_3_SessNo;
Int32 Blk_3_UserID;
char Blk_3_LMode;
char Blk_3_OType;
Int32 Blk_3_ObjDBId;
Int32 Blk_3_ObjTId;
char Blk_3_Status;
char MoreBlockers; /* Indicates there are more lock conflicts */
Int16 LogonSourceLen; /* Logon source length (Variable Length) */
char LogonSource[128]; /* Logon source info (Variable Length) */
} MonSesRec2_t;
typedef struct
{
    double TempSpaceUsage; /* Temporary space usage for the associated session */
    fltreal64_t HotAmp1CPU;
    fltreal64_t HotAmp2CPU;
    fltreal64_t HotAmp3CPU;

```

```

    fltreal64_t HotAmp1IO;
    fltreal64_t HotAmp2IO;
    fltreal64_t HotAmp3IO;
    Word HotAmp1CPUId;
    Word HotAmp2CPUId;
    Word HotAmp3CPUId;
    Word HotAmp1IOId;
    Word HotAmp2IOId;
    Word HotAmp3IOId;
    fltreal64_t LowAmp1CPU;
    fltreal64_t LowAmp2CPU;
    fltreal64_t LowAmp3CPU;
    fltreal64_t LowAmp1IO;
    fltreal64_t LowAmp2IO;
    fltreal64_t LowAmp3IO;
    Word LowAmp1CPUId;
    Word LowAmp2CPUId;
    Word LowAmp3CPUId;
    Word LowAmp1IOId;
    Word LowAmp2IOId;
    Word LowAmp3IOId;
    Word AmpCount;
    fltreal64_t AvgAmpCPUsec;
    fltreal64_t AvgAmpIOCnt;
} MonSesRecV3_t;
typedef struct
{
    double ReqStartTime;
    Int32 ReqStartDate;
    fltreal64_t ReqCPU;
    fltreal64_t ReqIO;
} MonSesRecV4_t;
typedef struct
{
    Int32 ReqNo;
    Int32 WlcId;
    Int16 DontReclassifyFlag;
} MonSesRecV5_t;
typedef struct
{
    fltreal64_t AvgCPU;
    fltreal64_t AvgDisk;
    fltreal64_t AvgDiskIO;
    fltreal64_t HighCPUUse;

```

```

    tospcnum_t    HighCPUProcId;
    fltreal64_t   LowCPUUse;
    tospcnum_t    LowCPUProcId;
    fltreal64_t   HighDisk;
    tospcnum_t    HighDiskProcId;
    fltreal64_t   LowDisk;
    tospcnum_t    LowDiskProcId;
    fltreal64_t   HighDiskIO;
    tospcnum_t    HighDiskIOProcId;
    fltreal64_t   LowDiskIO;
    tospcnum_t    LowDiskIOProcId;
    fltreal64_t   NetUse;
    char          NetAUp;
    char          NetBUp;
    Integer       ResLogging;
    Integer       ResMonitor;
    char          Release[29];
    char          Version[32];
} MonPhySum_t;
typedef struct
{
    fltreal64_t   AMPAvgCPU;
    fltreal64_t   AMPAvgDisk;
    fltreal64_t   AMPAvgDiskIO;
    fltreal64_t   HiCPUAMPUse;
    tospcnum_t    HiCPUAMPNo;
    tospcnum_t    HiCPUAMPProc;
    fltreal64_t   LoCPUAMPUse;
    tospcnum_t    LoCPUAMPNo;
    tospcnum_t    LoCPUAMPProc;
    fltreal64_t   HiDiskAMP;
    tospcnum_t    HiDiskAMPNo;
    tospcnum_t    HiDiskAMPProc;
    fltreal64_t   LoDiskAMP;
    tospcnum_t    LoDiskAMPNo;
    tospcnum_t    LoDiskAMPProc;
    fltreal64_t   HiDiskAMPIO;
    tospcnum_t    HiDiskAMPIONo;
    tospcnum_t    HiDiskAMPIOProc;
    fltreal64_t   LoDiskAMPIO;
    tospcnum_t    LoDiskAMPIONo;
    tospcnum_t    LoDiskAMPIOProc;
    fltreal64_t   PEAvgCPU;
    fltreal64_t   HiCPUPEUUse;

```

```

    tospocnum_t    HiCPUPENo;
    tospocnum_t    HiCPUPEProc;
    fltreal64_t    LoCPUPEUse;
    tospocnum_t    LoCPUPENo;
    tospocnum_t    LoCPUPEProc;
    fltreal64_t    SessionCount;
    Integer        SesMonitorSys;
    Integer        SesMonitorLoc;
    Integer        ResLogging;
    Integer        ResMonitor;
    char           Release[29];
    char           Version[32];
} MonVirSum_t;
typedef struct
{
    char           NetAUp;
    char           NetBUp;
    Int16          SampleRate;
} MonVirRes1_t;
typedef struct
{
    char           NetAUp;
    char           NetBUp;
    char           SystemType[7];
} MonConfig_t;
typedef struct
{
    Integer        ProcId;
    Integer        VProcNo;
    char           VprocType[3];
    Int16          HostId;
    char           Status;
    Int16          DiskSlice;
} MonVirConfig_t;
typedef struct
{
    Integer        ProcId;
    char           Status;
    char           CpuType[7];
    Integer        CpuCount;
} MonPhyConfig_t;
typedef struct
{
    char           VprocType[3];

```

```

Integer      ProcId;
Integer      VProcNo;
Integer      ClusterNo;
fltreal64_t  CPUUse;
char         Status;
Integer      SessLogCount;
Integer      SessRunCount;
fltreal64_t  DiskUse;
fltreal64_t  DiskReads;
fltreal64_t  DiskWrites;
fltreal64_t  DiskOutReqAvg;
fltreal64_t  HostBlockReads;
fltreal64_t  HostBlockWrites;
fltreal64_t  MemAllocates;
fltreal64_t  MemAllocateKB;
fltreal64_t  PctService;
fltreal64_t  PctAMPWT;
fltreal64_t  PctParser;
fltreal64_t  PctDispatcher;
fltreal64_t  NetReads;
fltreal64_t  NetWrites;
fltreal64_t  NVMemAllocSegs;
} MonVirRes2_t;
typedef struct
{
    Integer      ProcId;
    Integer      AmpCount;
    Integer      PECCount;
    fltreal64_t  CPUUse;
    fltreal64_t  PercentKernel;
    fltreal64_t  PercentService;
    fltreal64_t  PercentUser;
    char         Status;
    fltreal64_t  NetAUse;
    fltreal64_t  NetBUse;
    fltreal64_t  DiskUse;
    fltreal64_t  CICUse;
    fltreal64_t  DiskReads;
    fltreal64_t  DiskWrites;
    fltreal64_t  DiskOutReqAvg;
    fltreal64_t  HostBlockReads;
    fltreal64_t  HostBlockWrites;
    fltreal64_t  SwapReads;
    fltreal64_t  SwapWrites;

```

```

    fltreal64_t    SwapDrops;
    fltreal64_t    MemAllocates;
    fltreal64_t    MemAllocateKB;
    fltreal64_t    MemFailures;
    fltreal64_t    MemAgings;
    fltreal64_t    NetReads;
    fltreal64_t    NetWrites;
} MonPhyRes2_t;
typedef struct
{
    Integer        NumOfSteps;
    Integer        CurStepBegNum;
    Integer        CurStepEndNum;
} MonSqlStepInfo_t;
typedef struct
{
    fltreal64_t    EstRowCount;
    fltreal64_t    ActRowCount;
    fltreal64_t    EstElapTime;
    fltreal64_t    ActElapTime;
} MonSqlStepStats_t;
typedef struct
{
    char           OldAcct[MAXNAME] ;
    Int16          ErrorCode;
} SetSessAcct_t;
typedef struct ERROR_FAIL_Type {
    Word           StatementNo;
    Word           Info;
    Word           Code;
    Word           Length;
    char           Msg[255];
} Error_Fail_t;
#define MAXFIELDS 100
typedef struct InfoPairs
{
    short FType;
    short FSize;
} InfoPairs;
typedef struct DataInfo_t
{
    short field_count;
    InfoPairs FieldData[MAXFIELDS];

```

```
} DataInfo_t;  
#pragma pack()  
#endif
```

# MONITOR SESSION Response Combinations

The following lists all the possible combinations of the PESTate and AMPState response values returned from a MONITOR SESSION request and the resulting effect of each combination on session states. For a detailed description of each of these response values, see [MONITOR SESSION](#).

## Sample PM/API MONITOR SESSION Request Responses

As described in the table below, the MONITOR SESSION request returns both the PESTate and AMPState response values. A client performance monitoring application interprets the meaning of the PESTate and AMPState combination, which is displayed as the session state in the client application.

You can view different session states in Teradata Viewpoint and filter for the types of sessions you wish to view.

### Note:

The session state is not returned by any PM/API CLIV2 or Teradata JDBC Driver request. It is interpreted by the client application.

| If PESTate is... | AND AMPState is... | THE resulting session state displays... |
|------------------|--------------------|-----------------------------------------|
| HOST-RESTART     | any AMP state      | HOST-RESTART                            |
| ABORTING         | ABORTING           | ABORTING                                |
| ABORTING         | ACTIVE             | ABORTING                                |
| ABORTING         | BLOCKED            | ABORTING                                |
| ABORTING         | IDLE               | ABORTING                                |
| ABORTING         | UNKNOWN            | ABORTING                                |
| PARSING-WAIT     | ABORTING           | ABORTING                                |
| PARSING-WAIT     | BLOCKED            | BLOCKED                                 |
| PARSING-WAIT     | ACTIVE             | PARSING                                 |
| PARSING-WAIT     | IDLE               | PARSING                                 |
| PARSING-WAIT     | UNKNOWN            | PARSING                                 |
| PARSING          | ABORTING           | ABORTING                                |
| PARSING          | BLOCKED            | BLOCKED                                 |
| PARSING          | ACTIVE             | PARSING                                 |



| If PESTate is...   | AND AMPState is... | THE resulting session state displays... |
|--------------------|--------------------|-----------------------------------------|
| PARSING            | IDLE               | PARSING                                 |
| PARSING            | UNKNOWN            | PARSING                                 |
| DISPATCHING        | ABORTING           | ABORTING                                |
| DISPATCHING        | BLOCKED            | BLOCKED                                 |
| DISPATCHING        | ACTIVE             | ACTIVE                                  |
| DISPATCHING        | IDLE               | IDLE                                    |
| DISPATCHING        | UNKNOWN            | ACTIVE                                  |
| ELICIT CLIENT DATA | ABORTING           | ABORTING                                |
| ELICIT CLIENT DATA | BLOCKED            | BLOCKED                                 |
| ELICIT CLIENT DATA | ACTIVE             | ACTIVE                                  |
| ELICIT CLIENT DATA | IDLE               | ACTIVE                                  |
| ELICIT CLIENT DATA | UNKNOWN            | ACTIVE                                  |
| RESPONSE           | ABORTING           | ABORTING                                |
| RESPONSE           | BLOCKED            | BLOCKED                                 |
| RESPONSE           | ACTIVE             | RESPONSE                                |
| RESPONSE           | IDLE               | RESPONSE                                |
| RESPONSE           | UNKNOWN            | RESPONSE                                |
| RESPONSE-HELD      | any AMP state      | RESPONSE-HELD                           |
| ACTIVE             | ABORTING           | ABORTING                                |
| ACTIVE             | BLOCKED            | BLOCKED                                 |
| ACTIVE             | ACTIVE             | ACTIVE                                  |
| ACTIVE             | IDLE               | ACTIVE                                  |
| ACTIVE             | UNKNOWN            | ACTIVE                                  |
| IDLE: IN-DOUBT     | ABORTING           | UNKNOWN                                 |
| IDLE: IN-DOUBT     | BLOCKED            | UNKNOWN                                 |
| IDLE: IN-DOUBT     | ACTIVE             | UNKNOWN                                 |
| IDLE: IN-DOUBT     | IDLE               | UNKNOWN                                 |
| IDLE: IN-DOUBT     | UNKNOWN            | UNKNOWN                                 |
| IDLE               | ABORTING           | IDLE                                    |

| If PESTate is... | AND AMPState is... | THE resulting session state displays...                                                                                                                                                                                                                                                    |
|------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IDLE             | BLOCKED            | IDLE                                                                                                                                                                                                                                                                                       |
| IDLE             | ACTIVE             | IDLE                                                                                                                                                                                                                                                                                       |
| IDLE             | IDLE               | IDLE                                                                                                                                                                                                                                                                                       |
| IDLE             | UNKNOWN            | IDLE                                                                                                                                                                                                                                                                                       |
| DELAYED          | any AMP state      | DELAYED                                                                                                                                                                                                                                                                                    |
| SESDELAYED       | any AMP state      | SESDELAYED                                                                                                                                                                                                                                                                                 |
| QTDELAYED        | any AMP state      | QTDELAYED                                                                                                                                                                                                                                                                                  |
| BLOCKED          | IDLE               | BLOCKED<br><b>Note:</b><br>A BLOCKED session state in monitor partition session means some background activity is in progress and the last request is on hold until this background activity is completed. Database locks are not directly involved in BLOCKED monitor partition sessions. |

## Additional Information

### Teradata Links

| Link                                                                                                    | Description                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="https://docs.teradata.com/">https://docs.teradata.com/</a>                                     | Search Teradata Documentation, customize content to your needs, and download PDFs.<br>Customers: Log in to access Orange Books.                                                                                                                            |
| <a href="https://support.teradata.com">https://support.teradata.com</a>                                 | One-stop source for Teradata community support, software downloads, and product information.<br>Log in for customer access to: <ul style="list-style-type: none"><li>• Community support</li><li>• Software updates</li><li>• Knowledge articles</li></ul> |
| <a href="https://www.teradata.com/University/Overview">https://www.teradata.com/University/Overview</a> | Teradata education network                                                                                                                                                                                                                                 |
| <a href="https://support.teradata.com/community">https://support.teradata.com/community</a>             | Link to Teradata community                                                                                                                                                                                                                                 |