

Teradata Vantage™ - Workload Management

User Guide

Release 17.10




July 2021

Copyright and Trademarks

Copyright © 2016 - 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.

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

Part I: Introduction to Workload Management	6
Chapter 1: Overview	7
Introduction to Workload Management	7
Changes and Additions	7
What is Workload Management?	7
Chapter 2: Workload Management Features and Benefits	9
Workload Management Features	9
Business Goals	10
Key Benefits	10
Chapter 3: System Tools and Architecture	11
Viewpoint Workload Designer	11
Teradata Dynamic Workload Manager	11
Viewpoint Workload Monitor	11
Viewpoint Workload Health	11
PM/APIs and Open APIs	11
TDWM DIP	11
QueryBand	11
Utilities	12
Interaction of Workload Management Components	12
Chapter 4: Preparing to Implement Workload Management	13
Determining Workload Management Needs	13
Enabling TASM in Viewpoint	14
Part II: Working with Workload Management	15
Chapter 1: Rulesets	16
Ruleset Basics	16
Working with Rulesets	17
Considerations for Rulesets	17
Key Recommendations for Rulesets	18
Chapter 2: Workloads	19
Workload Basics	19
WD-Default Workload	19
WD-MapsMover Workload	19

Workload Priority Management	19
Virtual Partitions	21
Tactical Tier	22
SLG Tier	22
Timeshare Tier	25
Classification Criteria	26
Service Level Goals	31
Minimum Response Time	32
Data Block Selectivity	33
Workload Evaluation Order	33
Session Workload Assignment	33
Workload Priority Order	34
Workload Definition Summary	34
Mappings for Console Utilities	35
Key Recommendations for Workloads	36
Chapter 3: Throttles	37
Throttle Basics	37
Throttle Types	38
Query Session Throttles	38
Request Throttles	39
Working with Classification Criteria for Throttles	48
Combining System and Workload Throttles	49
Key Recommendations for Throttles	49
Chapter 4: Utility Management	50
Utility Management Overview	50
Supported Utilities	50
Utility Protocols	51
Utility Throttles	52
AWT Resource Limits	57
Utility Session Rules	61
Key Recommendations for Utilities	63
Chapter 5: Workload Exceptions	64
Exception Management	64
Exception Criteria Options	64
Tactical Workload Exceptions	65
CPU Disk Ratio	66
Skew Exception Criteria	66
Synchronous and Asynchronous Exception Checking	67
Exception Actions	67
Key Recommendations for Workload Exceptions	70
Chapter 6: Filters	71

System Filters	71
Key Recommendations for Filters	72
Chapter 7: Arrival Rate Meters	73
Arrival Rate Meters	73
Chapter 8: States and Events	77
States and Events Overview	77
State Matrix Basics	77
Events	80
Event Types	80
Key Recommendations for States and Events	93
Chapter 9: General Parameters	94
General View	94
Key Recommendations for General Parameters	101
Chapter 10: Advanced SQL Engine Analytic Functions and Workload Management	103
TASM with Advanced SQL Engine Analytic Functions	103
Creating a Viewpoint User for Data Collection	104
Adding an Advanced SQL Engine to the Monitored Systems Portlet	105
Example: Creating a TASM Ruleset to Limit SQLE Analytic Function Concurrency	105
Checking Query Status	106
Part III: Managing Workloads	109
Chapter 1: Monitoring Workload Management Activity	110
Using Viewpoint Workload Monitor	110
Using Viewpoint Workload Health	110
Chapter 2: Using Log Files	111
About the Log Files	111
DBQLogTbl	111
TDWMEventLog	111
TDWMSummaryLog	111
TDWMEventHistory	111
TDWMEceptionLog	113
Workload Activity Logging and DBQL Logging Comparison	113
Appendix A: TASM and TIWM Features	114
Appendix B: Additional Information	115



Introduction to Workload Management

Overview

Introduction to Workload Management

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 document provides information about Teradata Workload Management, which enables database administrators (DBAs) to manage workloads and direct how Teradata Vantage accomplishes expectations. This document is intended as an overview for technical users. For advanced recommendations for implementing workload management, see [Teradata® Active System Management Orange Book](#).

Changes and Additions

Date	Description
July 2021	Added information on the Arrival Rate Meter feature. See Arrival Rate Meters .
June 2020	Made the following changes for this release: <ul style="list-style-type: none">• Explained that queries that access Native Object Store tables consume unusually large amounts of memory.• Added Advanced SQL Engine Analytic Functions and Workload Management.

What is Workload Management?

A workload is a class of database requests with common traits whose access to the database can be managed with a set of rules. Workloads are useful for:

- Setting different access priorities for different types of requests.
- Monitoring resource usage patterns, performance tuning, and capacity planning.
- Limiting the number of requests or sessions that can run at the same time.

Workload management is the act of managing Vantage workload performance by monitoring system activity and acting when pre-defined limits are reached. Workload management uses rules, and each rule applies only to some database requests. However, the collection of all rules applies to all active work on the platform.

Users can employ Teradata Workload Management as a performance tool to direct how Vantage accomplishes expectations.

Teradata Workload Management offers two different strategies. See your Teradata representative for details about the license required for each strategy.

Teradata Active System Management (TASM)

TASM performs full workload management.

TASM gives administrators the ability to prioritize workloads, tune performance, and monitor and manage workload and system health. TASM automates tasks that were previously labor-intensive for application DBAs and operational DBAs.

Teradata Integrated Workload Management (TIWM)

TIWM provides basic workload management capabilities to customers without full TASM.

TIWM offers a subset of TASM.

While this document touches upon TIWM, it focuses primarily on TASM. For more information on the differences between TASM and TIWM, see [A Comparison of TASM and TIWM Features](#).

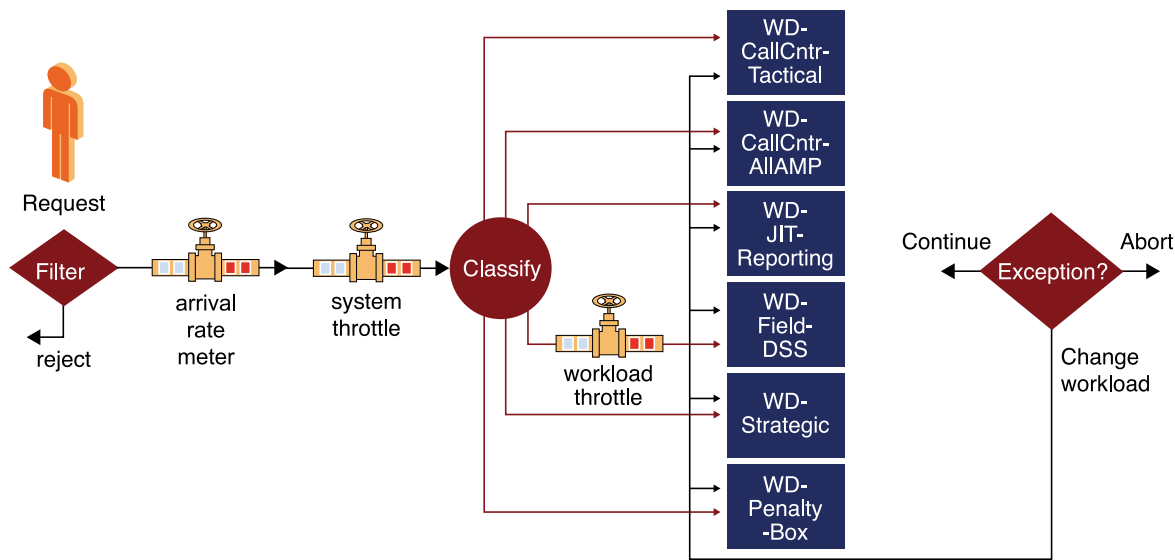
Workload Management Features and Benefits

Workload Management Features

TASM provides many ways to manage workloads, including the following features:

- Filters reject requests with specified characteristics before they start executing. Filters are applied system-wide.
- Arrival Rate Meter (ARM) regulates the flow of requests into the system.
- System throttles manage request concurrency across all workloads in the system. System throttles may cause the system to delay some requests when throttle thresholds are exceeded.
- Session throttles limit the number of sessions that can log on at the same time.
- Workload throttles manage request concurrency within one workload. Workload throttles may cause the system to delay some requests when throttle thresholds are exceeded.
- Classification defines the characteristics that qualify a request to run under the rules of this workload and determines which workload manages a specific request. Classification criteria may include--but are not limited to--originating application, target database object, or request characteristics.
- Events monitor and take action on utilization and activity.
- States allow TASM to enforce different rules in different circumstances, depending on system health and time frame.
- Utility management controls and customizes utility behavior separately from other request activity.
- Priority management controls the amount of CPU and I/O resources that individual requests receive, as defined by workload rules.
- Exception management detects unexpected situations within a workload and can automatically act to change the workload the request is subject to, or sends a notification.

The following shows the ways in which TASM manages workloads.



Business Goals

Data warehouses today run a complex mix of applications, all competing at once for database resources. Frequent data loading competes with business-critical tactical requests and strategic requests for business decision-making. Each workload type has different system resource needs and user response time expectations. TASM brings order to this environment. DBAs can use TASM to specify Service Level Goals for selected workloads, monitor performance toward those goals, and define actions for Vantage to take to avoid slowdowns of critical work.

Specifically, users create TASM rules that automate how work submitted to Vantage executes under different circumstances and when the database takes the following actions:

- Limits sessions, utilities, and requests – delaying or rejecting them when limits are reached
- Divides system resources among different types of requests
- Sends alerts or changes the workload of active requests that exceed limits
- Monitors time of day and system resources and reacts in predefined ways

Key Benefits

Teradata workload management provides the following benefits:

- Stabilizes response times of important work
- Prioritizes and protects known, proven work from poorly written requests
- Prioritizes work based on business operations and rhythms
- Manages system resources by type of work or department
- Manages resource use
- Offers techniques to manage unexpected situations

System Tools and Architecture

Viewpoint Workload Designer

The Viewpoint **Workload Designer** portlet enables users to create rules for managing the system workload.

Teradata Dynamic Workload Manager

Teradata Dynamic Workload Manager (TDWM) consists of code within Vantage to manage incoming work according to user-created rules. The core of TDWM is in the Dispatcher portion of the DBS.

Viewpoint Workload Monitor

The Viewpoint **Workload Monitor** portlet enables users to monitor workload activity in Vantage.

Viewpoint Workload Health

The Viewpoint **Workload Health** portlet displays workload management activity in Vantage for one system at a time and enables users to monitor the status of individual workloads.

PM/APIs and Open APIs

PM/APIs and open APIs (also called SQL interfaces) provide CLIV2 and SQL functions to access workload management and system information. Performance Monitor and Production Control (PMPC) data is collected internally and used to monitor workload management events. Several Viewpoint portlets use PM/APIs to monitor system activity. For more information on PM/APIs and open APIs, see *Teradata Vantage™ - Application Programming Reference*, B035-1090.

TDWM DIP

The Database Initialization Program (DIP) script DIPTDWM creates the database infrastructure for workload management or migrates existing workload management rules to another release or platform.

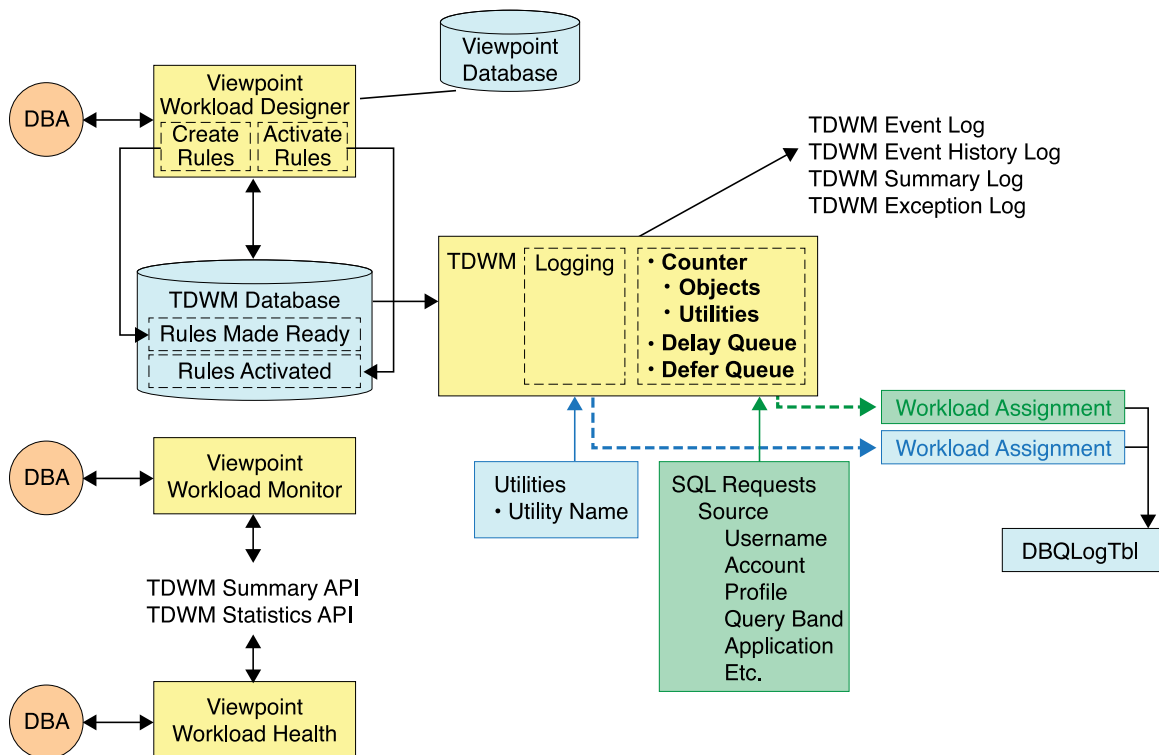
QueryBand

A query band is a set of name-value pairs assigned to a session, transaction, or profile. A query band helps users identify the originating source of the request. Query bands are particularly useful for identifying specific users submitting requests from a middle-tier tool or application.

Utilities

Teradata utilities for loading, exporting, and BAR also place demands on Vantage. TASM must balance utility operations with user requests.

Interaction of Workload Management Components



Preparing to Implement Workload Management

Determining Workload Management Needs

Take the following actions before implementing workload management.

Action	Additional Information
Monitor single system metrics and resource usage trends within specified time frames with the Viewpoint Metric Heatmap portlet.	For information on the Metric Heatmap portlet, see <i>Teradata® Viewpoint User Guide</i> , B035-2206.
Study resource contention at various times of day and to understand blocking and wait time patterns with the Viewpoint Query Monitor and Lock Viewer portlets.	For more information on the Query Monitor and Lock Viewer portlets, see <i>Teradata® Viewpoint User Guide</i> , B035-2206.
Enable DBQL detailed logging for at least 3 weeks to understand request performance. <ul style="list-style-type: none"> • Capture DBQLogTbl, DBQLSQLTbl, and DBQLObjTbl data. • After detailed logging captures enough information, use the LIMIT THRESHOLD or LIMIT SUMMARY options for highly-tuned short work. 	
Enable ResUsage.	ResSPMA, ResSPS, and ResSAWT are extremely helpful with system, scheduler, and AMP Worker Task (AWT) analysis. The ResUsageSPS table can help you determine resource usage by workload.
Identify workload types, such as ad hoc, ETL, and short highly-tuned requests: <ul style="list-style-type: none"> • Determine how these workloads are identified on the system; for example, by account strings, profiles, and groups of users. • Identify different processing periods, such as day, night, weekday, and weekend. • Determine the priorities given to different workload types during different processing periods. 	
Document all Service Level Goals.	
Document the exceptional conditions that now require the DBA to intervene, such as long request runtime, high request CPU use, high skew, and too many sessions or requests.	



Action	Additional Information
Document current operational limits, such as optimal concurrency levels for load utility jobs and all application groups.	
Ensure the Viewpoint server software is current.	

Everyone involved with the Teradata system needs to understand and agree about business priorities. Many workload management performance issues are caused by user disagreements about the relative importance of the workloads.

Examples: Basic Workload Definitions

Workload	Description	Time Period	Priority
Business Users	BI reporting and data mining	8 a.m. - 5 p.m. Monday-Friday	Low
Special Reporting	High-priority requests	8 a.m. - 5 p.m. Monday-Friday	High
Ad hoc SQL	Investigative decision-support requests	8 a.m. - 5 p.m. Monday-Friday	Medium
Daily Loads	Daily loading process	1 a.m. - 5 a.m. Daily	Medium
Data Maintenance	Statistics, archiving, and purges	Weekend	Low

Enabling TASM in Viewpoint

1. Ensure you have TASM.
For more information, contact your Teradata Account Manager.
2. From the Viewpoint portal page, select .
3. Open the **Monitored Systems** portlet.
4. Next to **Systems**, select , then select **Add Teradata System**.
For complete configuration instructions, see *Teradata® Viewpoint User Guide*, B035-2206.
5. Under **Enhanced TASM Functions** in the **General System Details**, select the check box if you have TASM. Do not select it if you have TIWM.



Working with Workload Management

Rulesets

Ruleset Basics

Rulesets are the foundation of workload management. Administrators use the **Workload Designer** portlet to create rules that define how the work submitted to Teradata should behave and under what circumstances. Rulesets control the following database behavior:

- Which requests are allowed to run
- How many requests with specific characteristics can be admitted into the system every hour/minute/second
- How many requests with specific characteristics can run at the same time
- What priority the requests should run in
- What Vantage does if a request behaves unexpectedly or uses too many resources
- How many system resources a request can consume

In a stable environment, one ruleset is all you need. After you create all rules in the main ruleset and activate it and test it, you might try out a new ruleset to see if it is an improvement. To do that, either activate the new ruleset or clone your existing ruleset to preserve original rules and then make changes. In either case, keep the old ruleset definition in case you need to return to it. Only one ruleset can be active at a time. Delete all rulesets you no longer need.

Note:

Do not change the ruleset to make periodic setting changes in TASM, for example, changing the ruleset when the system is very busy. TASM state changes accommodate that need with minimal overhead. Ruleset changes involve greater overhead and are appropriate only when you are setting up a new system or tuning an existing one.

New SLES 11 and Teradata Data Warehouse appliances are delivered with the FirstConfig ruleset enabled. FirstConfig has basic throttle rules that ensure that users cannot overrun the system with requests. You will want to create a new ruleset adapted to the specific needs and goals of your business.

Examples: Simple Rules

Rule Name	Attributes	Current State
Filter Financial_Tbl	Non-Finance users are not allowed to access Financial_Tbl	Enabled
Throttle DaveFields	Session limit on user DaveFields	Enabled limit = 3
WD Tactical	Classify by Single-AMP into a Workload Definition (WD)	Enabled

Rule Name	Attributes	Current State
WD Short	Classify by Est Time < 10 seconds into a Workload Definition	Enabled
WD Long	Classify by Est Time > or equal to 10 seconds into a Workload Definition	Enabled
WD Queries	Classify by Account = Finance into a Workload Definition Exception if CPU skew > or equal to 20 Exception action: Change to WD Long	Throttle 10 concurrent Enabled
Exception Disk Ratio	If CPU Disk Ratio > 10 Action: Abort on SELECT	No workloads qualify

Working with Rulesets

Use the following **Workload Designer** portlet features to manage rulesets. The Teradata Viewpoint administrator can grant your role the Edit Rulesets privilege, which is required to complete all of the following actions except lock and unlock a ruleset.

Workload Designer Feature	Description
Create a new ruleset	You can create multiple rulesets, but only one ruleset is active on Vantage at a time. After creating a ruleset, use the toolbar to specify settings, such as states, sessions, and workloads. A new ruleset is automatically locked so only the owner can edit it.
Edit a ruleset	You can edit rulesets only in the Working section.
Clone a ruleset	Makes an exact copy of the ruleset, except for the name. Cloning is a convenient way to create a ruleset using an existing ruleset as a basis.
Make a ruleset ready	Send an inactive ruleset from Viewpoint to Vantage.
Make a ruleset active	Activate the ruleset on Vantage.
Delete a ruleset	Removes the ruleset and all associated information.
Import and export a ruleset	Copies a ruleset from one Viewpoint system to another. You can import only rulesets exported from the Workload Designer portlet and a database of the same release.
Lock and unlock a ruleset	Locks a ruleset to prevent others from changing it. Only the lock owner or a user with permission to unlock any ruleset can unlock a ruleset.

For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

Considerations for Rulesets

For normal operating procedures, Teradata recommends that sites rely on a single ruleset. However, you can create varied states to define how TASM allocates resources during different times of the day, different days of the month, or when external conditions or system-wide events degrade system performance. Moving from one state to another, due to planned or unplanned circumstances, results in an automatic change in the workload management settings. Only one state in one ruleset can be in effect at a time. For more information on states, see [States and Events Overview](#).

Within a ruleset, each rule (specifically, session control, filter, system throttle, or Workload Definition) is divided into fixed attributes and working values. Fixed attributes do not change when the system state changes; however, working values for a rule can change to meet the needs of a particular state or planned environment.

Key Recommendations for Rulesets

Teradata recommends the following best practices for rulesets:

- Work with only one ruleset for simplicity and to maintain consistent IDs in data logged to DBQL and ResUsage
- Create a new ruleset only if the first ruleset needs major changes
- Remove old rulesets when their purpose becomes obsolete or unknown

Workloads

Workload Basics

When a request begins, TASM classifies it into a workload. A workload is a group of incoming requests that have similar characteristics, such as a source (application or user group), a request type (tactical versus strategic), or a resource expectation (estimated processing time). Administrators can monitor and adjust workloads to meet user expectations.

WD-Default Workload

WD-Default is the default workload Vantage applies to requests that are not assigned a workload. You can manage work in WD-Default with workload throttles and TASM workload exceptions. You cannot delete or disable the default workload.

WD-MapsMover Workload

WD-MapsMover is a system-defined workload dedicated to reassigning tables from one map to another. The default priority of this workload is Timeshare High, but the workload may be moved to any other Timeshare access level. To specify that a MAPS Mover session execute in this workload, use this query band name-value pair: `WDClassification=MoveTableToMap`. If you use Viewpoint to reassign tables to maps, Viewpoint sets this query band automatically.

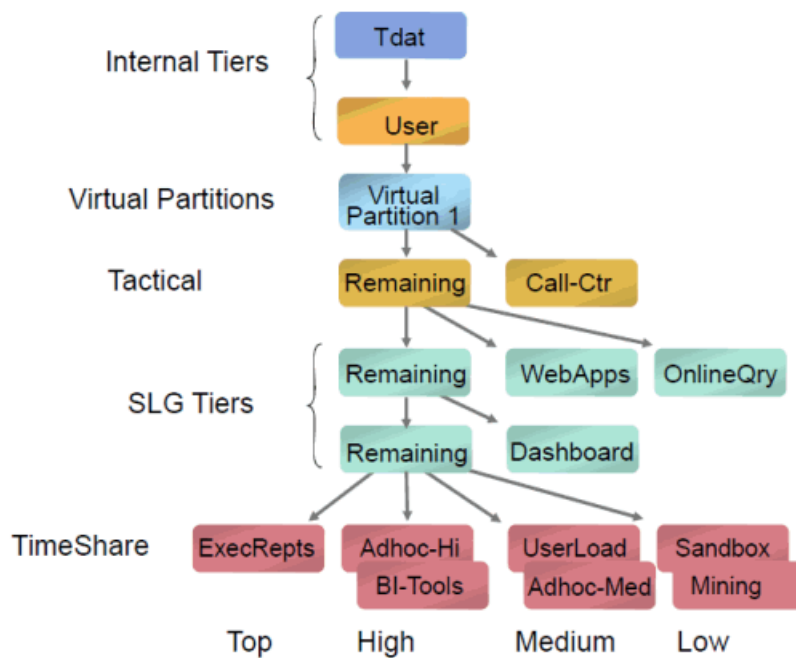
Workload Priority Management

When you define a workload for a Linux SLES system, the **Workload Designer** portlet prompts you to choose a workload management method. Select one of the following workload management methods to give a workload its place in the resource consumption hierarchy:

- Tactical
- SLG (up to 5 optional tiers)
- Timeshare (Top, High, Medium, and Low tiers)

TASM uses these tiers to prioritize requests in a workload and, optionally, to control when they can begin to execute.

The following figure illustrates the priority hierarchy levels.



Tactical Tier Workload Management Method

The Tactical tier is for workloads with critical, very short requests that need a response in 1 second or less. Tactical tier requests receive the highest priority available to user work and can consume all the resources they need. Teradata recommends that you use the Tactical tier primarily for single-AMP or few-AMP requests.

Workloads in the Tactical tier run with an expedited status automatically, which gives requests a performance boost by ordering them higher in the AMP message processing queues and by providing access to reserved AMP worker tasks.

SLG Tier Workload Management Method

Note:

This workload management method is available only on systems with TASM.

The optional SLG tier is for the following types of workloads:

- All-AMP tactical
- Those whose requests have service level goals
- Non-tactical applications whose response time is critical

Timeshare Tier Workload Management Method

The Timeshare tier is intended for workloads whose response time is less critical to the business and that do not have service level expectations. Use the Timeshare tier for the following types of workloads:

- Requests without a Service Level Goal
- Medium- or low-priority requests
- Background requests
- Sandbox applications

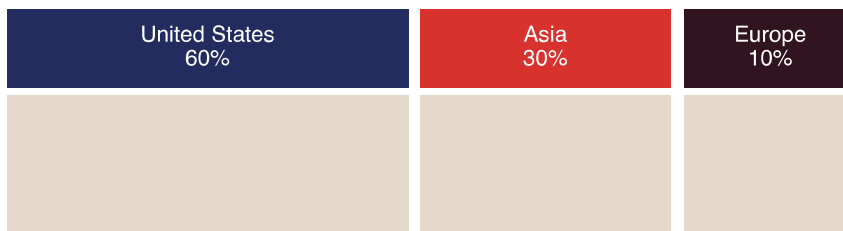
Timeshare tier workloads are intended to consume the majority of system resources.

Virtual Partitions

Note:

Only one virtual partition is available on systems with TIWM.

In the priority hierarchy, the first level that an administrator can interact with is the virtual partition. A virtual partition represents a collection of workloads. The default is one virtual partition, but you can define up to 10. One virtual partition is adequate to cover the needs of most systems, and is a good starting point. Multiple virtual partitions are intended for platforms that support distinct business units or geographic entities that need to be kept separated.



Each virtual partition offers Tactical tier, SLG tiers, and Timeshare tier for workload placement within the virtual partition.

Note:

There is one set of health conditions and planned environments that span all virtual partitions. Only the workloads are placed into different virtual partitions.

Virtual Partition Resource Allocation

Each virtual partition must be allocated a percentage of system resources. If there is only one virtual partition, it is automatically allocated 100% of system resources. You can change virtual partition allocation values for each planned environment.

Dynamic and Fixed Virtual Partitions

Virtual partitions are allocated a system resource percentage that is, by default, a dynamic (soft) limit. If other virtual partitions are not using their full allocation, then a busy virtual partition can consume more than it is allocated. In contrast, if the system resource percentage is fixed, the virtual partition cannot consume more than allocated, even if other virtual partitions do not use their full allocation. Use the **Workload Designer** portlet to specify whether a virtual partition allocation is dynamic or a fixed percent.

Workload priority is influenced by the virtual partition resource allocation and whether it is dynamic or fixed. Tactical requests running in a virtual partition with a low resource allocation may not perform as well as those in a virtual partition with a high resource allocation, particularly if the allocation percentage is fixed.

Tactical Tier

Tactical tier workloads are the first level under the virtual partition in the priority hierarchy. TASM gives this tier all the resources it needs, and the remaining resources are passed down to the next level. Tactical tier is for urgent requests that need results immediately. This tier is optional. Because Tactical tier requests can consume unlimited resources, be careful when assigning workloads there.

Consider the following guidelines when deciding whether to assign a workload to the tactical tier:

- Assign only workloads that are highly tuned and very short, primarily involving only a few AMPs or one AMP.
- Do not assign workloads that support load utilities.
- Monitor exceptions regularly and adjust the exception thresholds as needed.
- Do not place tactical workloads in virtual partitions with an inadequate resource allocations.

Performance Enhancements for the Tactical Tier

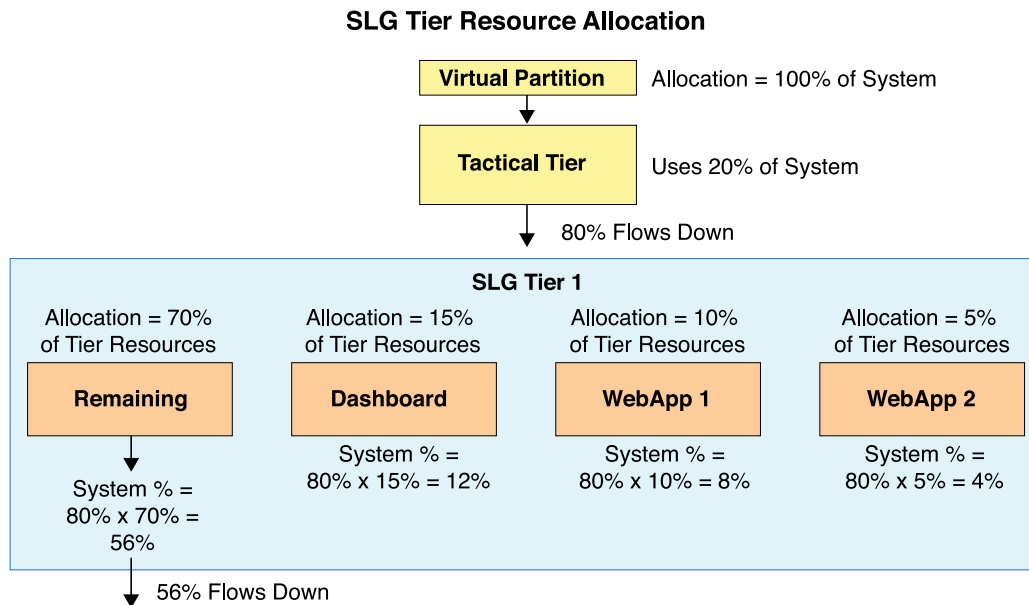
The critical workloads running in the Tactical tier receive additional performance support so that they can complete requests quickly. Two performance enhancements help work in this tier meet expectations:

- Expedited access to AMP message processing queues
- Reserved AMP worker tasks

Tactical tier workloads use these enhancements by default.

SLG Tier

Under the Tactical tier in the priority hierarchy, remaining resources flow down to the SLG tier. You can define up to 5 SLG tiers. Each workload in an SLG tier is assigned a percentage of tier resources to ensure that requests perform consistently and meet Service Level Goals. Unused resources flow down each of the SLG tiers with assigned workloads, in SLG tier order, and then on to the Timeshare tier. For example, in the following figure, workload WebApp1 is allocated 10% of tier resources. If the percentage of resources flowing into the tier is 80% of node resources, then WebApp1 is allocated only 8% of node resources.



Requests in an SLG tier workload divide resource allocations equally. For example, if the workload is allocated a 5% share and there are five requests running concurrently, each request gets 1% of the shared resources.

To accommodate growth, Teradata recommends that you assign workloads a higher allocation percentage than you think is needed. If workload consumption varies over time, allocate the workload a different percentage for each planned environment. Do not relocate workloads to different SLG tiers in different planned environments.

If there is more than one SLG tier, the allocation percentages assigned to workloads on the higher tiers translate to a larger share of resources than those on the lower tiers. For example, SLG tier 1 workloads receive service ahead of SLG tier 2 workloads.

The unused resources flowing down from an SLG tier to the tiers below equals 100% minus the sum of the workload allocation values on that tier. In the preceding figure, the sum of the three workload allocation values on the tier is 30%. This guarantees that at least 70% of the resources flowing into that tier are available to workloads in the tiers below.

See the Orange Book, *Teradata Priority Scheduler for Linux SLES 11*, 541-0008867 for guidance on how to set up SLG tier workloads.

Add Hard Limits to a Workload

You can, optionally, apply hard limits to SLG tier workloads. A hard limit is a percentage of *system* resources (CPU and I/O) that the workload can consume. This limit is independent from the share of *tier* resources the workload has. You cannot apply hard limits to Tactical or Timeshare tier workloads.

The system enforces workload hard limits exactly as you define them in the **Workload Designer** portlet. You do not need to multiply a workload hard limit by the workload management capacity on demand (WM

COD) percent, unlike virtual partition hard limits that do factor in the WM COD system limit. This makes it easy to monitor any workload with a hard limit.

If both virtual partition and workload hard limits exist, each is enforced independently. However, the workload hard limit is not enforced if that workload is under a virtual partition with a fixed allocation that is less than the workload hard limit.

Note:

Use hard limits with caution. Overuse of hard limits can cause system resources to be unused.

Expedited Workloads

Expedited workloads have expedited access to AMP message processing queues, similar to a fast pass at an amusement park. Expedited workloads are intended for the following types of requests.

Tier	Request Type
Tactical	Well-tuned single-AMP or few-AMP
SLG tier 1	Short, critical all-AMP

Considerations for Use of Expedited Workloads on the SLG Tier

Tactical workloads are automatically expedited. Expediting is optional for workloads in SLG Tier 1. Before you expedite an SLG tier 1 workload, make sure that the requests running in that workload meet the following criteria:

- They consume resources moderately.
- They do not hold onto AMP worker tasks so long that they compete against Tactical requests also using the reserved AMP worker task pool.

Use DBQLogTbl to analyze this. It provides the workload ID and the start and end times of each request that it logs.

- They are adequately tuned.

If too much work is expedited, it weakens the advantage that expedited status has on very short, highly critical work. Being over-generous with this option creates new performance issues and imbalances. Consider expedited status only for truly exceptional cases, not for all the workloads running on SLG Tier 1.

Reserved AMP Worker Tasks

Requests need AMP worker tasks. If all of the AMP worker tasks on an AMP are in use, high-priority requests might have to wait for low-priority requests to release their AMP worker tasks. To protect high-priority requests from this scenario, in the **Workload Designer** portlet, use the **Reserved AWTs**

option, located in the **Limits/Reserves** tab within the **General** view. Reserved AMP worker tasks can boost the following types of expedited requests on a busy system:

- Tactical tier
- SLG Tier 1, workloads with the Enable SLG Expedite option selected

For more information on reserved AMP worker tasks, see the Orange Book, *Teradata AMP Worker Tasks and ResUsage Monitoring*, 541-0009643.

Timeshare Tier

Timeshare tier workloads are at the bottom of the priority hierarchy. The Timeshare tier has 4 priority levels to which workloads are assigned:

- Top
- High
- Medium
- Low

How many resources a Timeshare tier request gets depends on the priority level of the workload it runs in. Higher-priority workloads receive more resources than lower-priority workloads.

Each priority level is assigned a rate, as follows.

Timeshare Name	Rate
Top	8
High	4
Medium	2
Low	1

These rates cannot be changed. The rate of 8 gives Top requests 8 times the resources as Low requests. High requests get 4 times the resources as Low requests, and Medium requests get 2 times the resources as Low requests. The number of requests in a Timeshare workload does not affect the rate. Even if you have only one request in Top and 10 in Low, Top requests still get 8 times the resources as Low requests.

Under normal situations, abundant resources flow down to Timeshare workloads. SLG tier workloads rarely need all the resources they are allocated. However, when critical work surges at the SLG tier level, SLG workloads may consume resources more fully, limited only by the allocation percentage set aside internally for Timeshare.

It is acceptable to give a Timeshare workload a different priority in different planned environments, for example, giving the Batch workload a Low priority in the Business_Hours planned environment and a Top priority in the Weekend planned environment. All of Timeshare is considered a single tier in the priority hierarchy.

Keep in mind the following placement guidelines for Timeshare workloads:

- For simple implementations, you can achieve effective priority differences even without SLG tier workloads using the Top, High, Medium, and Low access levels.
- If you are not using SLG tiers, place workloads in Top if they are very high priority but do not qualify as tactical.
- Keep the **Timeshare Decay** option turned off for more predictable priority differentiation in Timeshare. For more information on the **Timeshare Decay** option, see [Timeshare Decay](#).

Classification Criteria

There is a common set of classification criteria used for workload, filter, and throttle rules.

Classification Criteria	Description
Request Source	<p>Request source comes from one of the following:</p> <ul style="list-style-type: none"> • Username • Account name (account string minus the first 4 characters) • Account string • Profile • Application • Client IP address • Client logon ID <p>Note:</p> <p>If a TASM rule contains multiple criteria of the same request source type (such as multiple users and multiple accounts), then only one of the types defined in the rule needs to match the attributes of the current request for the request to qualify for the rule. If the rule contains criteria of different request source types, then the attributes of the current request must match one of each of the types defined in the rule to qualify for the rule.</p> <p>Note:</p> <p>If a user and a profile are defined, you can select whether to AND or OR the user and profile criteria for the rule, since profiles are a collection of users.</p>
Request Target	<p>Which database objects the request operates on and which type of step applies to that object:</p> <ul style="list-style-type: none"> • Database • Table (excluding volatile tables) • View • Macro • Stored procedure • User-defined functions (this selection includes table operators) • User-defined methods • QueryGrid server <ul style="list-style-type: none"> ◦ The external server name, such as a Hadoop server, which the syntax accesses, for example: <code>SELECT * FROM Table1@Hadoopserver1;</code> ◦ The server object is extracted from the foreign table referenced (Table1 in the preceding example).

Classification Criteria	Description
	<ul style="list-style-type: none"> After you select target criteria, you can, optionally, select more specific criteria, such as “unconstrained product join against table XYZ”. See the following section <i>Adding More Specific Criteria for Request Targets</i>. <p>Note: In order for stored procedure requests to qualify for any rule, the rule must have stored procedure inclusion criteria. If a rule does not have stored procedure inclusion criteria, stored procedures do not qualify for that rule.</p> <p>Note: For the purpose of classification, all target objects are considered to be of the same type and will be OR'd together (except for a QueryGrid server, which is treated as a unique DBS object type).</p>
Query characteristics	<p>Processing details:</p> <ul style="list-style-type: none"> Statement type (for example, DDL, DML, SELECT, COLLECT STATISTICS) Multistatement requests, see Multistatement Requests AMP limits (include only single or few-AMP requests or all-AMP requests) Step estimated row count Final estimated row count Estimated processing time Estimated step processing time Estimated memory usage, see Estimated Memory Usage Full-table scan (This criterion cannot be used for views) <p>Note: It is a best practice to use full-table scan instead as a more specific, secondary criterion of the specified target object. See the following section <i>Adding More Specific Criteria for Request Targets</i>.</p> <ul style="list-style-type: none"> Join type (all or no joins, all or no product joins, all or no unconstrained product joins) <p>Note: It is a best practice to use join type instead as a more specific criteria of the specified target object. See the following section <i>Adding More Specific Criteria for Request Targets</i>.</p> <ul style="list-style-type: none"> Incremental planning and execution (IPE) attribute, see Incremental Planning and Execution
Utility criteria	Which utility issues the request. This criterion is not available for system throttles.
Query band	The originating application may assign query band name/value pairs to a request. Use of the query band enables requests from a common logon to be classified into different workloads.

Note:

When you specify multiple qualification criteria, TDWM treats all of the specified criteria as ANDed conditions. The exceptions to this include DBS objects of the same kind and query bands with the same name.

Adding More Specific Criteria for Request Targets

After you create **Target** classification criteria, you can, optionally, select more specific classification criteria for most database objects by doing the following things in Viewpoint **Workload Designer**:

- Selecting or editing the **Target** criteria. An **Edit Target Criteria** dialogue box appears.
- Selecting the pencil icon in the **Selected:** area. An **Edit Criteria** dialogue box appears, where you can add more specific classification criteria.

Adding more specific classification criteria lets you pinpoint the exact characteristics of requests that qualify for a rule. For example, you can specify that a request qualifies for a rule if it operates on the specified table AND requires a product join on that table. Adding more specific classification criteria ensures that TDWM applies rules the way you intend. Consider the following request:

- In one step, a small table will require a join.
- In another step, there is no join, but a large table will generate many rows.

In this situation, you specify classification criteria for the following items:

- The large table
- Any joins
- Step row count

If you want TASM to detect if the large table returned many step rows AND was joined, the preceding request should not qualify for the rule, yet it may be a false positive. The only way to get the functionality you want is to specify **Any Join** and **Estimated Step Row Count** as more specific criteria for the large table. This means that you want both the join and the step row count to occur in any step in a request that operates on the large table.

You can apply the following more specific criteria to most DBS objects that are request targets:

- **Full Table Scan** (Include or Exclude)
- **Join Type** (all or no joins, all or no product joins, all or no unconstrained product joins)
- **Estimated Step Row Count** (above or below a specified number)
- **Estimated Step Processing Time** (above or below a specified number)

Note:

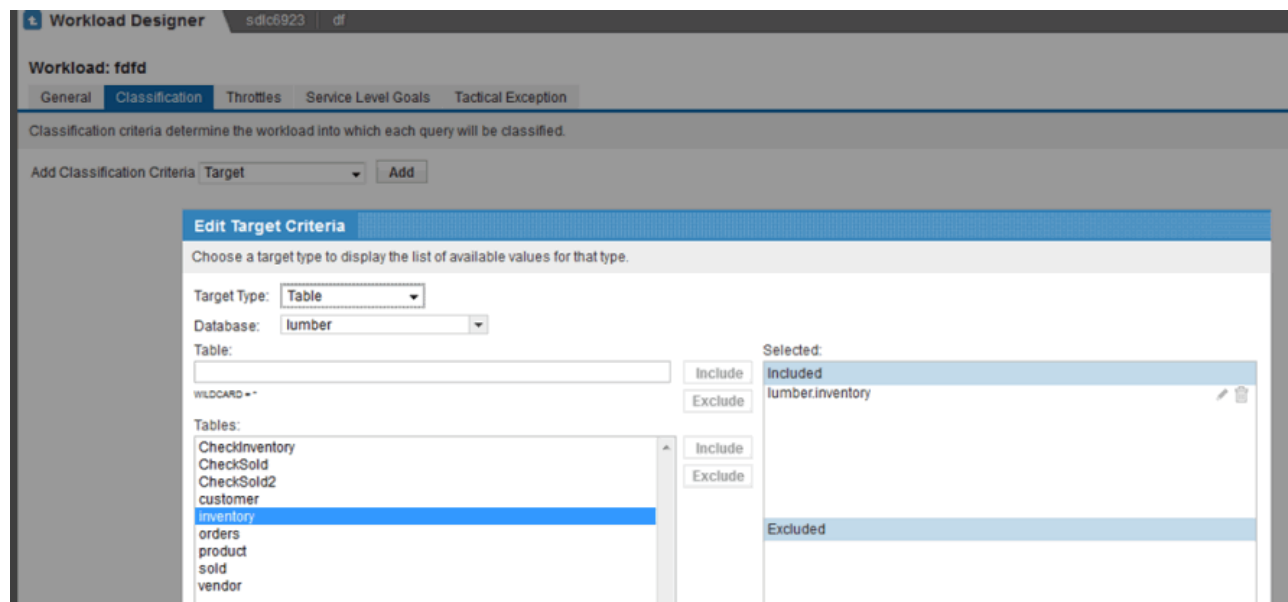
You cannot create more specific classification criteria for UDFs and UDMs.

You can specify the following more specific classification criteria for tables:

- **Estimated percent of table blocks accessed during the query** (greater than or equal to a specified percentage); this is the percentage of the table the request is expected to access

- **Only statements of the following type:**(DDL, DML, Select)

The following figure is an example of the **Edit Target Criteria** screen. To add more specific classification criteria to the inventory table, you would select the pencil icon in the **Selected:** area next to lumber.inventory.



Wildcards

You can use two wildcard characters anywhere in character strings representing DBS object names:

- The “*” wildcard character means match zero or more subsequent characters
- The “?” wildcard character means accept any legitimate character at this place in the string

Note the following examples:

- User names “ABC*” means all users with names beginning with “ABC” (including the name “ABC”)
- User names “A*C” means all users with names beginning with “A” and ending with “C” (including the name “AC”)
- Table names “DB1.T?BLE*” means all tables in database DB1 with names beginning with “T_BLE” where the second character could be any character
- Stored procedures “*.*” means all stored procedures in all databases

You can use multiples of these wildcard characters. For example:

- User names “A*B*C” means all users with names beginning with “A” and ending with “C,” with a “B” somewhere in the middle
- Table names “DB1.T???E” means all tables in database DB1 with names consisting of 5 characters beginning with “T” and ending with an “E”

Note:

Wildcards cannot be applied to volatile tables.

Estimated Memory Usage

Vantage logs estimated and actual memory use for each request step in DBQLogTbl. Use TASM to identify requests that use large amounts of memory, then classify or throttle these requests appropriately. XML functions, some aggregations, hash joins, and queries that reference Native Object Store tables consume unusually high levels of memory.

Teradata recommends that you use **Memory Usage** primarily for throttles.

You can specify **Memory Usage** as a criterion for a throttle in the **Throttles** view, **Classification** tab, **Query Characteristics** classification criteria. The choices for **Memory Usage** are:

- Increased
- Large
- Very large

The range of memory associated with each category is defined in the default memory usage thresholds for your platform configuration. To see the defaults, look under **System Settings** in Viewpoint Workload Designer.

Note:

The request memory estimate represents peak use, typically by the largest step.

Incremental Planning and Execution

Incremental Planning and Execution (IPE) is a request processing method designed for complex requests. In IPE, Vantage breaks requests into smaller pieces and then optimizes each piece one at a time. The results from earlier request pieces help Vantage plan how to process later request pieces.

Dynamic Plan

Because Vantage can "learn" from the results of earlier request pieces, an IPE plan is called dynamic.

Static Plan

A static plan optimizes requests as a single unit. A static plan may provide some IPE functionality, but TDWM will work from the static plan.

Vantage applies IPE automatically to qualifying complex requests.

You can use IPE as a classification criterion for workload, filter, and throttle rules. This allows you to identify IPE requests being executed and manage them differently from non-IPE requests. You can use

the NumFragments field in DBC.DBQLogTbl to identify IPE requests. NumFragments reports the number of fragments (pieces) for IPE requests. This field is null for non-IPE requests.

Because an IPE dynamic plan gives TASM only a partial view of a request, TASM ignores the following step-level criteria when attempting to classify an IPE request:

- Step estimated row count (minimum or maximum)
- Step estimated time (minimum or maximum)
- Full-table scan
- Join type
- IPE request criteria

Multistatement Requests

Business intelligence tools, such as Tableau, group multiple SQL statements into one request. The **Multi-Statement Request** classification criterion lets you identify, isolate, and manage multistatement requests using the following options:

Option	Description
Exclude	Prevents multistatement requests from being included in a specific workload.
Include	Splits multistatement requests into a distinct workload so you can manage them separately from shorter requests.
Threshold statement count	Enter a value to define the minimum number of SQL statements that qualify as a multistatement request. The default is 2.

This classification criterion applies to workload, filter, and throttle rules and can be combined with all existing classifications.

TASM ignores the following special statements when counting the statements in a request:

- BEGIN TRANSACTION (BT)
- END TRANSACTION (ET)
- COMMIT
- NULL

The TDWMMSRCount field in DBC.QryLogV view shows the statement count in a multistatement request.

Service Level Goals

Note:

This feature is available only on systems with TASM.

Important workloads often come with service level goals to indicate their performance objectives. This is especially true for tactical workloads. TASM lets you set goals on workloads to reflect the needs of the business.

Service level goals are measurable using either of the following metrics:

Response Time

The percentage of completed requests that meet the response time goal (for example, 2 seconds or less 99% of the time)

Throughput

Requests completed per hour (for example, 1,000 requests per hour)

Service level goals can trigger system health conditions via the workload events for **SLG Response Time** or **SLG Throughput**. You can also use service level goals in reports, such as using `DBQLogTbl.ResponseTimeMet`.

Note:

Service level goals are most appropriate for Tactical and SLG tier workloads.

Minimum Response Time

To manage user expectations about request response times, you can specify a minimum response time for a workload. TASM prevents requests from responding before the specified time. By delaying the return of their answer sets, minimum response times protect users from developing unrealistic expectations in situations where fast response times are temporary, such as on a lightly loaded, recently upgraded system. In Viewpoint, configure the minimum response time on the **Hold Query Responses** tab when you create or edit a workload.

Another scenario in which minimum response times are useful is when capacity is added to a system. Without a minimum response time, some users could see a big boost in response speed and start submitting more work. The extra work would consume capacity ahead of plan. A minimum response time would hold response speed to historical expectations and prevent users from loading the system with less important work.

Minimum response times are primarily useful for requests with Service Level Goals, where consistency is expected and users notice and care about elapsed times. Minimum response times are not available for Tactical workloads.

Minimum Response Time Characteristics

Note the following specifics about minimum response time:

- TASM can hold responses from 1 to 3,600 seconds, and the value can vary by planned environment.
- The minimum response time does not affect SET commands, EXEC commands, stored procedure calls, and transaction statements. For stored procedures, TASM does not hold the CALL statement,

but it treats each statement in the procedure as a separate request. This means that each request can have a minimum response time if the request is in a workload with a minimum response time. This is also true for statements in a macro.

- TASM places a request on hold at the point where the database would normally send the response to the client: AMP steps have completed, locks are released, and TASM throttle counters have been decremented.
- The RESPONSE-HELD PE state, which is shown in the Viewpoint **Workload Monitor** portlet, indicates that a response is on hold. TASM can abort a request in the RESPONSE-HELD state. However, TASM cannot release a held request until the minimum response time expires.
- Once the return of an answer set has been placed on hold, the minimum response time for the request does not change due to planned environment or rule set changes.

Data Block Selectivity

Use the data block selectivity criterion (the percentage of the table the request accesses) to give more resources to requests that access a small subset of the table. For example, you can send a request doing a full-table scan against a large table to a low-priority workload, and send a request accessing only one partition (1/200th of the table) to a high-priority workload.

Data block selectivity is particularly useful on large partitioned tables, but it is also useful for nonpartitioned tables.

Workload Evaluation Order

When TASM classifies a request, it may match more than one workload. To avoid uncertainty as to which workload a request should be assigned to, adjust the workload evaluation order. When multiple workloads could apply to a request, the system always uses the workload closest to the top of the list.

Consider evaluation order carefully. The classification process stops when the request finds a match. In the following scenarios, evaluation order affects results in surprising ways:

- If you have a workload with a criterion of Not-All-AMPs because you want Tactical requests to be evaluated first, utility work may unexpectedly go in the workload. This is because utility work can sometimes use only some AMPs.
- If you include a criterion of USER = * early in the evaluation order, all requests would run there, and subsequent workloads would not be considered.

Tip:

Configure workload evaluation order so that it starts with criteria that is most specific (such as DBA, BAR, and load utility work) and ends with criteria that is most generic. This ensures that work classifies to the proper workload.

Session Workload Assignment

When a session logs on, Vantage assigns it to a workload. This workload is used for session management and parser-related activity for the life of the session. The session workload is logged in the SessionWDID field in DBC.DBQLogTbl.

When assigning a session workload, Vantage considers only the request source criteria of the workload (such as user and account name). The workload assigned to a session is the first workload in workload priority order whose request source criteria matches the logon information of the session.

Next, after Vantage parses a request, it classifies each request into a workload using all classification criteria and the plan created by the parser. The request can be assigned to a different workload than the session. Requests that use only the parsing engine, such as CALL, SHOW, and HELP statements will run in the session workload.

Workload Priority Order

TASM prioritizes workloads based on the workload management method, along with the tier and allocation percent for SLG tiers or the access level for Timeshare. TASM uses the workload priority order to assign the session workload and, optionally, to order requests in the delay queue. The workload management methods and tiers or access levels in the following table are shown in descending order of priority.

Workload Method	Tier or Access Level
Tactical	
SLG Tier	1
	2
	3
	4
	5
Timeshare	Top
	High
	Medium
	Low

Note:

By default, TDWM orders requests in the delay queue **By time delayed**. **By workload priority** is another choice for the **Order the Throttle Delay Queue** option. This option is available in the **General** area of Viewpoint **Workload Designer** on the **Other** tab.

Workload Definition Summary

Workload Characteristics	Required	Optional	Description
Classification	Yes		Which requests should be assigned to the workload.
Throttle		Yes	How many requests in this workload can run concurrently.
Priority	Yes		The priority assigned to the workload. Advanced SQL Engine with SLES 11 has a hierarchy-based priority management system.
Service Level Goals		Yes	Service level goals (SLGs) for workload request performance, defined as either response time or throughput. For example, queries in the WD-Dashboard workload are expected to complete in 2 seconds, while requests in WD-MktRepts are expected to complete within 1 hour.
Exceptions		Yes	Requests that start to execute and then display characteristics unsuitable for the workload, such as high skew or too much CPU processing. Note: Tactical exceptions and Tactical exception actions are required for Tactical workloads.
Exception Actions		Yes	The automatic actions to take when an exception occurs.
Hold Query Responses		Yes	Minimum response times for requests. This option provides a greater degree of consistency within a workload, whether the system is busy or lightly loaded.
Virtual Partitions		Yes	A collection of workloads. One virtual partition is adequate for most systems. Platforms that support distinct business units or geographic entities may use multiple virtual partitions to keep this work separated. Each virtual partition supports Tactical, SLG tier, and Timeshare workload management methods.

Mappings for Console Utilities

Console utilities, such as ScanDisk and CheckTable, do not log on or behave like SQL requests. For example, CheckTable does not log on, so there is no user associated with a CheckTable job. Because of these differences, TASM does not assign console utilities to workloads automatically. The **Console Utilities** tab on the **Workloads** view of Viewpoint **Workload Designer** allows you to assign console utilities to the appropriate workload.

Console Utility to Workload Mapping

Lists each console utility, and you can specify a workload to apply it to. By default, the console utility runs in that workload.

Performance Group to Workload Mapping

Lists each legacy performance group (H, L, M, and R), and you can choose the workload name to apply it to. The console utility runs in this workload when the user sets the utility to run in the performance group using the priority statement. For example Priority = H.

These settings have the following order of precedence with respect to console utilities:

- If a priority is specified, the utility runs under the workload specified for that priority in **Performance Group to Workload Mapping**.
- If no priority is specified, the utility runs under the workload specified in **Console Utility to Workload Mapping**.
- If no workload is specified, the utility runs under the **WD-Default** workload.

Under most circumstances, console utilities are considered Timeshare Low or Timeshare Medium work. The Query Session console utility, however, is given medium or high priority because it is interactive. The increased priority may be beneficial because DBAs often use Query Session to look at active sessions when a system is resource-constrained.

Key Recommendations for Workloads

The goal when creating a workload is to efficiently manage system work and minimize workload administration. To achieve this balance, use the following guidelines:

- Create a workload only when there is a clear need (5-20 workloads are enough for most sites).
- Simplify workload classification wherever possible so that the effects are clear.
- Classify workloads first by request source or query band because these criteria are the most exact and predictable. Many customers use the request source option **Profile**, which categorizes work according to environment and role.
- Use target criteria, request characteristics, and utility criteria mainly as secondary classification criteria to achieve further differentiation between request sources and query bands.

Throttles

Throttle Basics

TASM must make choices on a busy system when demand for resources exceeds availability. Throttles are TASM rules that limit the number of requests, sessions, or load utilities that can run at the same time. Throttles help improve the performance of all requests by reducing resource contention. If a throttle threshold is exceeded, TASM either rejects the work or delays it until the system is less busy. Throttles do not affect work that is already running.

You can apply throttles to the following targets:

- The entire system
- Specific workloads or groups of workloads
- Virtual partitions
- Utility jobs

System throttles can be active without workload throttles, and vice versa. TASM can also use system and workload throttles together. One request could be under the control of both a system throttle and a workload throttle.

Use workload throttles primarily to prevent too many low-priority requests from running at once, leaving no resources for higher-priority requests.

Examples: Throttles

Some common targets of throttles include the following types of work:

- Low-priority requests
- Requests that consume many resources
- Sandbox applications
- Full-table scans of medium-sized or large tables or Native Object Store table

An all-amp full-table scan keeps the AMP Worker Tasks occupied until it completes. If you run many of these requests concurrently (the number supported depends on your system configuration), other work may be delayed. This applies for all full-table scans, including Native Object Store tables.

Examples of throttles include the following TASM actions:

- Limit a specific user (source classification) to having no more than 10 sessions at a time (request session limit).
- Limit the number of application X sessions (source classification) simultaneously executing against a table (target classification).

- Allow no more than 3 concurrent data mining requests (source classification) because data mining uses significant resources and is not a high priority.
- Limit requests based on characteristics, for example, statement type, step row count, or join properties.
- Limit the number of requests that can run during end-of-the-month processing (**Limits by State**).

Throttle Types

TASM can apply throttles to the following:

Utilities

The **Utility Limits** throttle limits the number of utilities that can run at the same time

Session Logons

The **Query Sessions** throttle limits the non-utility sessions that can log on at the same time

Concurrent Requests

The number of requests that can be logged on at the same time can be throttled at the following levels:

- Entire system
- Specific workload
- Group of workloads
- Virtual partition (useful when multiple entities share a system)

These throttles are available in the **Workload Designer** portlet as follows.

Viewpoint Workload Designer Tab	Throttle Type
Sessions	Query Sessions
	Utility Limits
Throttles	System
	Virtual Partition
	Workload Group
Workloads	Workload

Query Session Throttles

The **Query Sessions** throttle limits the non-utility sessions that can log on at the same time. The **Query Sessions** throttle can only reject sessions trying to log on; it cannot delay session logons. Once TASM rejects a throttled session, the user has to log on again.

The **Query Sessions** throttle can use only source classification criteria. For example, it can include account names, profiles, or IP addresses. It may not include target criteria, such as a database name or table name.

Choose from the following four **Rule Type** options when you create a **Query Sessions** throttle:

Arrival Rate Monitor

Defines an admission limit as a rate that is an integer per time unit (for example, 10 requests/minute), instead of a simple integer. Each incoming SQL request is checked against arrival rate monitor (ARM) rules before throttle rules. When an applicable ARM limit is reached, the incoming request is deferred in a new queue, called the Defer. When the deferred request is released from the Defer queue (or admitted by TASM), it is checked against applicable throttle rules.

DIPTDWM must be run to create or migrate TDWM tables for ARM support.

Collective

Provides the least granular control. TASM treats all requests that qualify for this throttle as one group. If the requests are delayed, the delay is based on the concurrency level of the entire group. For example, users Jay, Kay, and May are the classification criteria for a system throttle rule with a limit of 3 requests. With the collective option, all the users together cannot have more than 3 requests running at the same time.

Individual

Each individual object that is associated with the throttle is treated independently of other, similar objects. Each object has a counter. For example, if users Jay, Kay, and May are the classification for a throttle rule with a limit of 3 requests, each of the users can have up to 3 requests running at the same time.

Member

Provides the most fine control. TASM treats each user in a throttled object (such as account) independently of other users in that account. The intent of the Member option is to enable objects that represent groups of users, such as accounts and profiles, to be used as the classification criteria for the rule. The rule specifies the group, and any member of the group is covered by the rule. For example, users Jay, Kay, and May are associated with account ShoeDept. If there is a system throttle rule with a session limit of 3, classification criteria of account ShoeDept, and the Member option, then each of the users with the ShoeDept account can have up to 3 requests running at the same time.

Request Throttles

The most popular use of throttles is to limit concurrent requests. When you enable a throttle rule, TASM counts the active requests the rule applies to. When a new request is ready to execute, TASM compares the request count with the rule limit. If the count is below the limit, the request runs immediately. If the count

is equal to or above the limit, TASM either rejects the request or puts it in a delay queue. Most throttles delay requests rather than reject them.

When system activity falls below the throttle limit, the system runs delayed requests in the order you select for releasing the throttle delay queue:

- **By time delayed:** The requests delayed the longest are released first
- **By workload priority:** The request from the highest-priority workload is released first

Note:

The **Order the Throttle Delay Queue** option is available in the **General** area of Viewpoint **Workload Designer** on the **Other** tab.

If a delayed request starts to run, it cannot be delayed again.

Note:

If a request is subject to multiple throttles, all throttles must be below the limit before the request can run.

System Throttles

System throttles control the concurrency of all requests, or a specified subset of requests, across the entire system. TASM rejects or delays requests controlled by system throttles. System throttles can use all classification criteria.

System throttles provide value when workload throttles are not suitable. System throttles are workload-independent and can manage requests across broader dimensions than workload throttles.

Choose from the following three **Rule Type** options when you create a system throttle:

Collective

Provides the least granular control. TASM treats all requests that qualify for this throttle as one group. If the requests are delayed, the delay is based on the concurrency level of the entire group. For example, users Jay, Kay, and May are the classification criteria for a system throttle rule with a limit of 3 requests. With the **Collective** option, all the users together cannot have more than 3 requests running at the same time.

Individual

Treats each user or object (such as a specific account) independently. If the throttle affects multiple objects, TASM counts each separately. If only one object is associated with this throttle, the individual and the collective types act the same. If a system throttle with users Jay, Kay, and May and a limit of 3 requests has the **Individual** option, then each of those users can have up to 3 requests running at the same time. When using the **Individual** option, you must specify objects of only one kind in the rule classification criteria. If there are

multiple objects types specified (for example, users and accounts or users and profiles), the **Collective** option is used. The **Individual** option does not support query band criteria.

Member

Provides the most granular control. TASM treats each user in a throttled object (such as account) independently of other users in that account. The intent of the **Member** option is to enable objects that represent groups of users, such as accounts and profiles, to be used as the classification criteria for the rule. The rule specifies the group, and any member of the group is covered by the rule. For example, users Jay, Kay, and May are associated with account ShoeDept. If there is a system throttle rule with a session limit of 3, classification criteria of account ShoeDept, and the **Member** option, then each of the users with the ShoeDept account can have up to 3 requests running at the same time.

Tip:

One way to remember the difference between these different rule types is: one, many, and many more.

- **One: Collective** is one throttle limit value, shared by the throttled objects.
 - **Many: Individual** can be many throttle limit values, one for each throttled object.
 - **Many more: Member** can be many more throttle limit values, depending on the throttled object and whether that object contains many other objects.
-

Disable Manual Release or Abort

If this option is selected, the DBA cannot release or abort a request in the delay queue, and TASM executes a delayed request only when all throttle limits that apply to the request are satisfied. Use this option only after careful consideration.

Throttling at the User Level

Use a member rule type system throttle to limit the concurrent requests of users with the same account or profile. This prevents a single user from unfairly dominating a workload, forcing other users to the delay queue.

Teradata recommends that you use a member type system throttle in conjunction with a workload throttle to ensure fairness. For example, create a system member throttle on Profile AcctUsers that is set to 1. Set the workload throttle for AcctUsers profile to 5. Then the AcctUsers workload is allowed 5 concurrent request (due to the workload throttle), but each user mapped to the AcctUsers profile may submit only 1 request (due to the system member throttle).

System Throttles and Stored Procedures

You must explicitly include a stored procedure name in the classification criteria if you want it to be covered by a system throttle. Otherwise, TASM ignores the stored procedure CALL statements and throttles only the SQL within the stored procedure.

TASM counts each time code calls a stored procedure. However, if a stored procedure calls itself, TASM does not count that. Consider the following example:

```
CALL ABC

ABC calls DEF

DEF calls DEF (recursive)
```

A throttle of 1 on stored procedure DEF will not impact this example.

Workload Throttles

TASM rejects or delays requests controlled by workload throttles.

Because workload throttles are predictable and easy to understand, Teradata recommends them as the main tool for limiting work on a busy system. Workload throttles are easier to monitor and make it easier to assess request delay times. It is also clearer which requests are affected by workload throttles. A system throttle often needs complex criteria to achieve the same scope as a workload throttle. This complexity can add confusion when analyzing throttle behavior and the impact of delays.

If SQL requests in a stored procedure classify to a throttled workload, TASM throttles the requests in the procedure individually.

Workload throttles work well when a hardware component is down, and the system is running in degraded mode. The administrator can reduce the throttle limit for low-priority workloads so that high-priority workloads can still deliver consistent response times.

Exceeded Throttle Limits

Sometimes the number of active requests exceeds throttle limits. This can happen for several reasons:

- A request uses too many resources, and TASM takes the exception action to move it to another workload. The destination workload is already at the active request limit. The moved request causes the workload to exceed the limit.
- A DBA moves an active request to a new workload. TASM has to override throttle limits to allow this action.
- A DBA forces a request in the delay queue to run immediately. TASM has to override throttle limits to allow this action.

- TASM changes the state, and the request limit for the new state is less than the limit for the prior state. The number of active requests exceeds the limit until some requests complete.
- TASM releases a delayed request that is causing a deadlock, even if the workload is already at the request limit.
- The DBA changes the ruleset. TASM counts active requests against the new ruleset, even if that causes the number of requests to exceed the new throttle limit.
- The **Prevent Mid-transaction Throttle Delays** option is enabled, and requests in a transaction execute when the throttle limit is reached.

Note:

It is also possible for throttle limits not to be met, even though there are requests in the delay queue, ready to execute, that are managed by that throttle. This can happen when a request is subject to multiple rules that perform throttling (including system throttles, workload throttles, and utility throttles), and one or more of them is already at the throttle limit. Because a request must satisfy the throttle limits for all applicable rules, if all rules cannot be satisfied, then the request is delayed.

When Workload Throttles Prevent Full Resource Usage

Workload throttles limit the number of requests that can run concurrently in a workload to prevent overutilization of system resources. However, in some cases, workload throttles can prevent full resource utilization. The **Enable Flex Throttles** option, available for SLES 11 systems with TASM for Enterprise Data Warehouse platforms, can detect situations in which workload throttle limits are preventing full resource utilization, and release work from the delay queue to use those resources. You can enable the flex throttles option for all states or enable and disable it by state.

When setting flex throttles, you define the following things:

- The **Flex Throttle Action Interval** (when there are available resources, how long TASM waits after releasing requests from the delay queue before repeating the action)
- The events that trigger flex throttle actions
- What happens when the triggering events occur

You can enable the flex throttles option in evaluation mode before activating it. In evaluation mode, no requests are released when the triggering event occurs. Instead, the TDWM event log lists which requests would have been released if the option had been activated.

Note:

The flex throttles option overrides only workload throttles. It does not override any of the following constraints:

- Object or system throttles
- Group throttles
- Utility throttle limits or AWT resource limits
- A workload throttle limit of 0

Note:

If you enable flex throttles, choose the default option for **Define Available AWTs** in the **Other** tab of the **General** view. Choosing the **AWTs available for the WorkNew (Work00) work type** option ensures that there are enough reserved AWTs to start the work flex throttles releases from the delay queue.

Enabling and Configuring Flex Throttles

1. Edit or create a ruleset.
2. From the ruleset toolbar, click **Throttles**.
3. In the **Workloads with Throttles Defined** section, click **Flex Throttles**.
4. Click the **Enable Flex Throttles** check box.
5. [Optional] To use the flex throttles in evaluation mode, select the **Enable Evaluation Mode** check box.
In evaluation mode flex throttles have no actual effect on your system, but the hypothetical results are reported in the TDWM event log.
6. Under **Triggering Events**, designate the **Available AWT** conditions to trigger an event.

Option	Description
Number of AMPs with Available AWTs	Specify the minimum number of AMPs with available AWTs. The event will be triggered if this number or more are available.
Number of Available AWTs	Specify the minimum number of available AWTs. The event will be triggered if this number or more are available.
Qualification Method	Specify the number of minutes for which <i>both</i> conditions above must be met to trigger the event.

7. [Optional] To further restrict the event definition to occur only when there is a minimum level of **CPU Utilization**, select the check box and configure the following:

Option	Description
System CPU	Specify the minimum system CPU utilization percentage. The event will be triggered if the CPU utilization is less than or equal to this value.
Qualification Method	Specify the number of minutes for which the minimum system CPU must be maintained to trigger the event. The average value of this metric must exceed the threshold for this time period.

8. [Optional] To further restrict the event definition to occur only when there is an I/O bottleneck, select the **I/O Usage** check box and configure the following:

Option	Description
Bandwidth	Specify the percentage of total I/O bandwidth that must be in use to trigger the event.
Monitored LUNs	TASM determines which LUNs are most vulnerable to throughput issues. Specify the percentage of these to monitor.
Triggered LUNs	The percentage of monitored LUNs that must be at the used Bandwidth percentage to trigger the event.
Qualification Method	Specify the interval at which an average Bandwidth is evaluated.
Qualification Time	Specify how long the condition must persist to qualify as an event.

9. Under **Actions**, enter the **Number of Queries to Release** on the flex throttle action interval. The number of queries cannot be more than twice the number specified for **Number of Available AWTs**. The flex throttle action interval is defined in the **Other** tab of the **General** category on the ruleset toolbar.
10. [Optional] Select actions to occur when a flex throttle event starts or ends.

Action	Description
Send Alert	Specifies the action to trigger.
Run Program	Specifies the Alerts registered programs to trigger.
Post to QTable	The string you enter is posted to the QTable.

11. Click **Save**.

Workload Group Throttles

Workload group throttles limit the number of requests that a group of workloads can run concurrently. To qualify for a workload group throttle, a workload must already have an individual throttle and must be active in at least one state. Before a request affected by a workload group throttle can run, it must satisfy both the workload throttle and the workload group throttle limits. A workload can belong to only one workload group throttle.

Note:

You can also use system throttles to enforce concurrency limits. Group throttles offer more flexibility in certain cases. Teradata expects that a just a few workloads per customer site will need to be grouped.

Example: Workload Group Throttles Limit Concurrency after Demotions

When TASM demotes requests to a lower-priority workload, the number of active requests can temporarily exceed workload throttle limits. However, workload group throttles can limit the number of active requests for an application, despite the unpredictable effect of workload demotions. The following examples describe the effect of demoted requests with different throttle types.

Demotions with Workload Throttles

In this example, you want to limit the concurrent requests from an application to 9. Two longer-running requests in the Medium workload cause an exception, and TASM demotes them from the Medium workload to the lower-priority Long workload. Both workloads have throttle limits, but if the Long workload throttle is already at the limit, it will exceed the limit temporarily when the demoted requests join the workload. Demoted requests are never delayed because they have already started to run. Also, after TASM demotes the 2 Medium requests, the Medium workload is below the throttle limit, so 2 new requests start running in that workload. Under these conditions, the total application requests can exceed what you intended.

Workload	Throttle Request Count	Throttle Request Count If TASM Demotes Two Medium Requests
Medium	6	6
Long	3	5
Total Active	9	11

Demotions with Workload Group Throttles

If there are many demotions in a short time and new requests replace the demoted requests in the higher-level workload, the total application requests continue to increase. In this situation, group throttles can keep the application request total below the limit.

In this example, a group throttle that combines Medium and Long is limited to 9 requests at a time. Although Medium loses 2 requests due to demotions, it cannot release 2 requests from the delay queue because of a group throttle. The group throttle is already at the limit of 9, so until 2 more requests affected by the group throttle complete, Medium stays below the throttle limit.

Workload	Throttle Request Count	Throttle Request Count If Two Medium Requests Are Demoted	Throttle Request Count If One Long Request Completes After the Medium Requests Are Demoted
Medium	6	4	5
Long	3	5	4
Group Throttle	9	9	9
Total Active	9	9	9

Examples: Workload Group Throttles Allow Capacity Sharing

If a workload is running fewer requests than the workload throttle allows, that workload is underused. With individual workload throttles, no other workload can benefit from this underused capacity. However, workload group throttles allow an underused workload and a busy workload running the same application to share capacity. In this situation, TASM divides requests between the workloads, and each workload has a workload throttle. The following examples describe the effect of an underused workload with different throttle types.

No Capacity Sharing with Workload Throttles

With individual workload throttles, if one workload is underutilized, no other workload can benefit from the unused capacity, even if both workloads run the same application. In this example, all requests from an application classify to one of two different workloads. Both workloads have throttles. If one of the workloads is not running all the requests it can, the entire application does not reach potential.

Workload	Request Count at Throttle Limit	Request Count If Only 3 Medium Requests Are Active
Medium	6	3
Long	3	3
Combined	9	6

Capacity Sharing with Workload Group Throttles

Group throttles allow a combination of throttled workloads to be treated as a group by a single, higher-level throttle. In this example, the application has a group throttle limit of 9 requests. To allow capacity sharing between the two workloads, each workload has a slightly larger throttle limit than the intended level. For example, Medium has a limit of 8 requests instead of 6. If one workload is not busy, the other workload can start more requests, up to the group throttle limit or the workload throttle limit.

Workload	Request Count at Throttle Limit	Request Count If Only 3 Medium Requests Are Active
Medium	8	3
Long	6	6
Application Workload Group Throttle	9	9
Combined	9	9

Virtual Partition Throttles

Virtual partition throttles control virtual partition resources when multiple entities share a platform. Priority allocations and hard limits control the CPU and I/O use in a virtual partition. However, those methods do not control other resources, such as AMP worker tasks. Virtual partition throttles cannot directly limit use of memory or AMP worker tasks. However, virtual partition throttles can have an indirect influence on their use.

A virtual partition throttle manages how many requests and utilities can run at the same time in the virtual partition. However, it does not count requests in Tactical workloads against the throttle limit.

When to Use a Virtual Partition Throttle

Use virtual partition throttles only as needed, and keep the TASM setup as simple as possible. Here are some situations in which virtual partition throttles can be useful:

- **Multi-tenant architecture:** Use virtual partitions to separate companies (tenants) who share a Teradata platform. In this situation, it is important to have strong use limits. Adding a virtual partition throttle ensures that one tenant cannot monopolize AMP worker tasks or other resources. When you set the throttle limit, consider the type of requests the tenant runs and how many AMP worker tasks their requests typically use (usually 1 to 4).
- **Geographic or business unit separation:** Virtual partitions are also useful when you need to separate business units or geographic locales. In this case, a virtual partition throttle may not be necessary. However, if there is a shortage of AMP worker tasks and if their use is not balanced evenly between virtual partitions, virtual partition throttles can help share resources more fairly.

When you are setting a virtual partition throttle limit, consider the type of work involved. If the work is dominated by complex requests with frequent parallel steps and redistributions or by many concurrent load jobs, a lower throttle limit may be needed. Consider a higher throttle limit for virtual partitions that do shorter, simpler work.

Working with Classification Criteria for Throttles

Throttles have the same classification criteria as filters and workloads. For a list of available classification criteria, see [Classification Criteria](#).

Combining System and Workload Throttles

A request could be affected by multiple throttles:

- A system throttle and a workload throttle
- Multiple system throttles

In either case, a request cannot run until all applicable throttles are below their limit.

Carefully consider the impact of multiple throttles. A request may be in the delay queue longer than expected if it gets released from one throttle and is then delayed by another.

Tip:

Teradata recommends that administrators keep throttles simple and avoid overlapping throttles. Overlapping throttles make it hard to analyze request behavior and tune throttle definitions.

Key Recommendations for Throttles

Consider the following guidelines for creating workload throttles:

- Set workload throttle limits first and then introduce system-level throttle limits later, if needed.
- Be cautious in applying throttles to Tactical workloads because they have a high priority and usually need to run immediately.
- Define workload throttle limits for both Base and system busy states.
- Favor production work by applying throttles to development work.
- Favor ad hoc and reporting work over administration requests.
- Throttle Low and Medium Timeshare workloads to avoid a shortage of AMP worker tasks.

When creating system throttles, explicitly exclude utilities. To do this, you can use the application name as a request source criterion.

Utility Management

Utility Management Overview

TASM can manage the following aspects of utilities:

- Utility sessions
- Concurrency limits
- Workload Definitions
- AWT resource limits

Supported Utilities

TASM can manage the following Teradata utilities.

Protocol	Utility Names
FastLoad	<ul style="list-style-type: none"> • FastLoad • JDBC FastLoad • Teradata Parallel Transporter Load operator • CSPLOAD (Crashdumps Save program Save Dump)
MultiLoad	<ul style="list-style-type: none"> • MultiLoad • Teradata Parallel Transporter Update operator
MLOADX	Teradata Parallel Transporter Update operator
FastExport	<ul style="list-style-type: none"> • FastExport • JDBC FastExport • Teradata Parallel Transporter Export operator
Backup/Restore	Data Stream Architecture (DSA)

Note:

The Teradata Parallel Data Pump (TPump) utility and the Teradata Parallel Transporter Stream operator use the SQL protocol. This means that Teradata manages them as SQL requests. The TASM utility management features described here do not apply to them.

Some third-party utilities use variations of the FastLoad, MultiLoad, and FastExport protocols. For utilities that implement the Teradata Parallel Transporter Application Programming Interface (Teradata Parallel Transporter API), TASM recognizes these utilities as the Load, Update, and Export operators of

Teradata Parallel Transporter. If a third-party utility implements other variations of these protocols, it is non-conforming and TASM may not be able to manage it.

Utility Protocols

Utilities use sessions differently. In general, a utility logs on the following types of sessions:

Control SQL Session

A session that executes SQL requests related to utility work. Almost all Teradata utilities have a control SQL session.

Auxiliary SQL Session

An optional, second SQL session maintains a restart log for recovery.

Utility Sessions

One or more sessions are used for sending and receiving data to and from Vantage.

The following table describes whether a utility uses these sessions.

Utility Names	Control SQL Session	Auxiliary SQL Session	Utility Sessions
FastLoad, MultiLoad, FastExport	Yes	Yes	Yes
Teradata Parallel Transporter Load, Update, and Export operators	Yes	Yes	Yes
MLOADX	Yes	Yes	Yes (using SQL sessions)
JDBC FastLoad, JDBC FastExport	Yes	No	Yes
CSP Save Dump	Yes	No	Yes
DSA	Yes	No	No

Utility sessions are subject to different throttles than SQL sessions. The SQL sessions and requests issued through them are subject to these throttles:

- Query session
- Workload
- System

Utility sessions are subject only to utility session rules.

TASM and Teradata utilities treat the whole load/unload process as one Unit of Work. A Unit of Work starts with the BEGIN LOADING/MLOAD/EXPORT request on the control SQL session. It ends with the END LOADING/MLOAD/EXPORT request. This means that a utility Unit of Work spans multiple SQL requests. Only the first SQL request of a unit on the control SQL session satisfies the utility classification criterion and is assigned to a utility workload. Subsequent requests are assigned to the same utility workload. SQL

requests outside the load/unload unit and SQL requests from the auxiliary SQL session can be assigned to different workloads. All applicable throttle counters are checked and incremented only at the beginning of a unit and decremented only at the end of the unit. For backup and restore, the entire script is treated as one unit.

Utility Throttles

Utility throttles limit the number of utilities that can run at the same time. A DBA can throttle utility concurrency at two levels.

Throttle Type	Utility Concurrency Limit	Viewpoint Tab
Utility Limits	System-level	Sessions
Workload	Workload-level	Workloads

A utility throttle includes the following values:

- Utility name
- Concurrency limit
- Delay/Reject option

Workload throttles have more qualification criteria, such as **Request Source**, **Target**, and **Query Band**.

TASM does not support the delay option for non-conforming utilities. TASM always rejects non-conforming utilities that exceed throttle limits, even when the throttle specifies **Delay**.

You might want to create utility throttles for the following reasons:

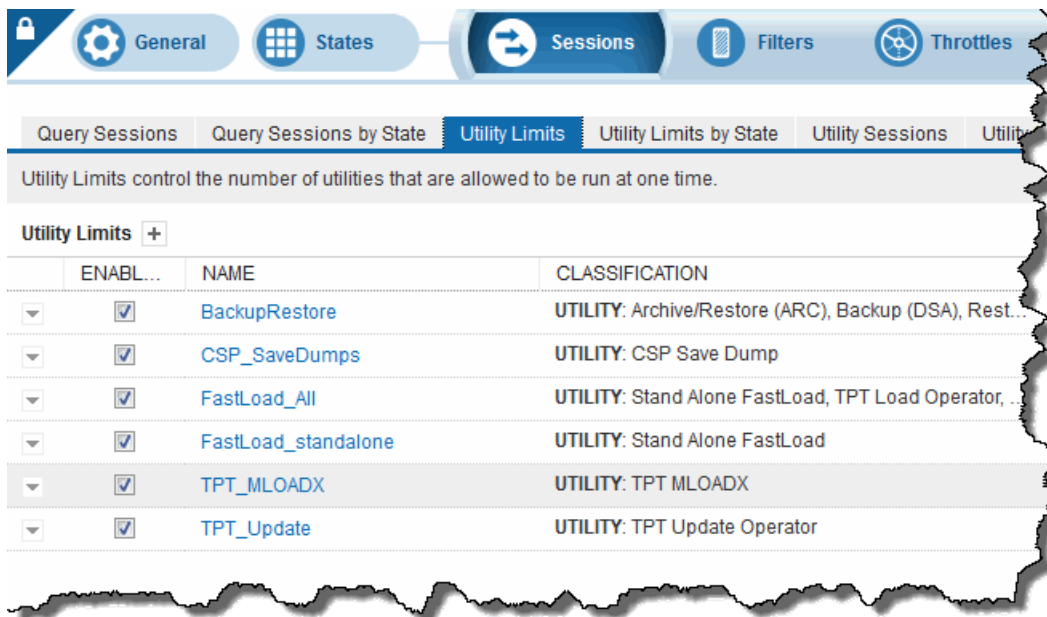
- To limit system resource use (for example, CPU, I/O, AMP worker tasks, and memory)
- To limit how many utilities can run in different system states, such as when the system is very busy

Examples: Utility Throttles

A DBA can create several utility throttles to limit different combinations of utilities, such as in these examples:

- Allow 10 load utilities of any kind with delay
- Allow 5 FastLoad utilities with delay.
- Allow 2 Teradata Parallel Transporter Updates with reject.
- Allow 15 DSAs with delay.
- Allow 40 MLOADX jobs with delay.
- Allow 5 DSA Backup jobs with delay.
- Allow 3 DSA Restore jobs with delay.

The following figure shows the system-level utility throttles.



Utility System Limits

In addition to user-defined utility throttles, TASM imposes the following utility system limits.

Protocols	Limits	Delay/ Reject	Utility Names
FastLoad+MultiLoad+FastExport	60	Delay	<ul style="list-style-type: none"> FastLoad, Teradata Parallel Transporter Load operator, JDBC FastLoad, CSP Save Dump, MultiLoad, Teradata Parallel Transporter Update operator, FastExport utility, Teradata Parallel Transporter Export operator, and JDBC FastExport Non-conforming FastLoad, MultiLoad, and FastExport utilities
FastLoad+MultiLoad	30	Delay	<ul style="list-style-type: none"> FastLoad, Teradata Parallel Transporter Load operator, JDBC FastLoad, CSP Save Dump, MultiLoad, and Teradata Parallel Transporter Update operator Non-conforming FastLoad and MultiLoad utilities
FastLoad	30	Delay	<ul style="list-style-type: none"> FastLoad, Teradata Parallel Transporter Load operator, JDBC FastLoad, and CSP Save Dump Non-conforming FastLoad utilities
MultiLoad	30	Delay	<ul style="list-style-type: none"> MultiLoad and Teradata Parallel Transporter Update operator Non-conforming MultiLoad utilities

Protocols	Limits	Delay/ Reject	Utility Names
MLOADX	30/120	Delay	Teradata Parallel Transporter Update operator
FastExport	60	Delay	<ul style="list-style-type: none"> FastExport utility, TPT Export operator, and JDBC FastExport Non-conforming FastExport utilities
Backup/Restore	350	Delay	Teradata DSA

Note:

- When a utility system limit is reached, **Delay** is the default if there is no user-defined utility throttle with a reject option.
- The default limit for MLOADX jobs is 30. You can get a higher limit of 120 if you select the option for **Support increased MLOADX job limits and increased AWT resource limits**. If you select that option, MultiLoad throttles do not apply to MLOADX jobs.

The following figure shows the **Support increased MLOADX job limits and increased AWT resource limits** option.

The screenshot shows the 'Throttles' configuration page in the Teradata Workload Management User Guide. The 'Other' tab is selected, and the 'Utility Limits' checkbox is highlighted with a red box. The 'Utility Limits' checkbox is checked, indicating that support for increased MLOADX job limits and increased AWT resource limits is enabled.

General **States** **Sessions** **Filters** **Throttles**

General **Bypass** **Limits/Reserves** **Other**

Intervals

Event Interval: 60 seconds

Dashboard Interval: 60 seconds

Logging Interval: 600 seconds

Exception Interval: 60 seconds

Blocker

Block Cycles: 1

Block Action: Release

Activation

Please select the features that will be enabled when this rule set is activated:

☒ Filters and Utility Sessions

☒ System Throttles and Session Control

Timeshare Decay

Timeshare decay automatically decreases the resource access rate of queries in a Timeshare workload over time.

☐ Enable Timeshare Decay

Prevent Mid-Transaction Throttle Delays

☐ Do not delay queries within transactions that are holding locks higher than 'access'

Order the Throttle Delay Queue

☒ By time delayed

☐ By workload priority

Utility Limits

☒ Support increased MLOADX job limits and increased AWT resource limits

Save Reset

Using System Throttles with Utilities

Teradata recommends that you do not use system throttles with utilities. When creating system throttles, explicitly exclude utilities. However, if system throttles do apply to utility work, make sure that the system throttle limit is greater than the utility throttle limit to prevent hung jobs.

Consider the following situation leading to a deadlock. There are a utility throttle and a system throttle that are applicable to the FastLoad utility. Both throttles are set at 2. When 2 FastLoad jobs are submitted, the following events occur:

- The first FastLoad starts a Unit of Work using the control SQL session associated with it. The system throttle counter is at 1.
- The second FastLoad starts a Unit of Work using the control SQL session associated with it. The system throttle counter is at 2.

- The first FastLoad tries to update the restart log on the auxiliary SQL session associated with it, but it is delayed because the system throttle counter is already at 2.
- The second FastLoad tries to update the restart log on the auxiliary SQL session associated with it, but it is delayed because the system throttle counter is already at 2.

Both FastLoads are delayed indefinitely.

Now consider a similar situation with a better result. In the following events, the system throttle is set to 3, greater than the utility throttle:

- The first FastLoad starts a Unit of Work using the control SQL session associated with it. The system throttle counter is at 1.
- The second FastLoad starts a Unit of Work using the control SQL session associated with it. The system throttle counter is at 2.
- The first FastLoad updates the restart log on the auxiliary SQL session associated with it. The system throttle counter is at 3, the limit. When the updates completes, the system throttle counter is back to 2.
- The second FastLoad updates the restart log on the auxiliary SQL session associated with it. The system throttle counter is at 3, the limit. When the updates completes, the system throttle counter is back to 2.
- The first FastLoad ends the Unit of Work using the control SQL session associated with it. The system throttle counter is at 1.
- The second FastLoad ends the Unit of Work using the control SQL session associated with it. The system throttle counter is at 0.

Utility Workloads

A utility workload includes **Utility** as a classification criterion. Other classification criteria for utility workloads include the following items:

- **Request Source**
- **Query Band**
- **Target** (not available for archive/restore utilities): **Database**, **Table**, and **View**

The following options are not available to utility workloads:

- Target classification criteria: macros and stored procedures
- Exceptions
- Query characteristics criteria

Tip:

Teradata recommends that you do not use Service Level Goals for utility workloads.

Utility Workload Throttles

Use utility workload throttles to define more detailed utility limits. For example, to allow user Becky to run no more than 2 FastExport utilities, do the following:

- Create a workload with the following classification criteria:
 - Username = Becky
 - Utility = Standalone FastExport
- Set the workload throttle to 2

Note:

Only the utility work classifies with this criteria, the corresponding auxiliary SQL session classifies to whichever workload a request from Becky would normally classify to.

If work from Becky normally classifies to a workload that would risk exposing the utility auxiliary SQL session to unacceptable delays, consider the following strategy:

- Creating a workload to classify utility auxiliary SQL work by the application name request source criteria. This workload would not have a throttle.
- Placing this workload in the workload evaluation order above where work from Becky would normally classify.

AWT Resource Limits

The purpose of the AWT resource limit is to prevent utility jobs as a group from using too many AMP Worker Tasks (AWTs) and possibly impacting non-utility work. Users can define their own AWT resource limits or use the TASM defaults. In the **Workload Designer** portlet, AWT resource limits are in the **Throttles** view under the **Resource Limits** tab. The limits are expressed as a percentage of maximum AWTs per AMP.

TASM settings override the DBS Control general fields MaxLoadTasks, MaxLoadAWT, MLOADXUtilityLimits, MaxMLOADXTasks, and MaxMLOADXAWT.

Default AWT Resource Limits

The value of the default AWT resource limits depends on the setting of the **Support increased MLOADX job limits and increased AWT resource limits** option on the **Other** tab of the **General** view in the **Workload Designer** portlet.

If you clear the option, TASM uses the following default AWT resource limits.

Utility Type	Percentage of the Total Number of AWTs per AMP Allowed for Utility Use
All load utilities as a group, in any combination (FastLoad, FastExport, MultiLoad, MLOADX)	60

Utility Type	Percentage of the Total Number of AWTs per AMP Allowed for Utility Use
DSA backup and DSA Restore	70

The AWTs per AMP are not reserved for utility use, but if they are available, utilities can use them up to the limit specified.

If you select the option, TASM uses the following AWT resource limits.

Utility Type	Percentage of the Total Number of AWTs per AMP Allowed for Utility Use
All load utilities as a group, in any combination (FastLoad, FastExport, MultiLoad, MLOADX)	70
MLOADX	70
DSA backup and DSA Restore	70

For example, if the default of 80 AWTs per AMP is in effect, then by default all load utilities combined can use up to 60% of 80, or 48 AWTs per AMP. If you check the MLOADX option, load utilities as a group can use 70% of AWTs per AMP, or 56 AWTs.

In addition, when you check the MLOADX option, TASM creates a separate limit of 70% of AWTs per AMP for MLOADX jobs and enforces it independently. Both the MLOADX limit and the load utility group limit must be satisfied before an MLOADX job can run.

The following figure shows the **Support increased MLOADX job limits and increased AWT resource limits** option.

The screenshot shows the 'Throttles' configuration page in the Teradata Workload Management User Guide. The page has a top navigation bar with tabs: General, States, Sessions, Filters, and Throttles. Below this, there are sub-tabs: General, Bypass, Limits/Reserves, and Other. The 'Other' sub-tab is selected. The page is divided into several sections: Intervals, Blocker, Activation, Timeshare Decay, Prevent Mid-Transaction Throttle Delays, Order the Throttle Delay Queue, and Utility Limits. The 'Utility Limits' section is highlighted with a red box and contains a single checkbox labeled 'Support increased MLOADX job limits and increased AWT resource limits', which is checked. At the bottom of the page, there are 'Save' and 'Reset' buttons.

General **States** **Sessions** **Filters** **Throttles**

General **Bypass** **Limits/Reserves** **Other**

Intervals

Event Interval: 60 seconds
 Dashboard Interval: 60 seconds
 Logging Interval: 600 seconds
 Exception Interval: 60 seconds

Blocker

Block Cycles: 1
 Block Action: Release

Activation

Please select the features that will be enabled when this rule set is activated:

☒ Filters and Utility Sessions
☒ System Throttles and Session Control

Timeshare Decay

Timeshare decay automatically decreases the resource access rate of queries in a Timeshare workload over time.

☐ Enable Timeshare Decay

Prevent Mid-Transaction Throttle Delays

☐ Do not delay queries within transactions that are holding locks higher than 'access'

Order the Throttle Delay Queue

☒ By time delayed
☐ By workload priority

Utility Limits

☒ Support increased MLOADX job limits and increased AWT resource limits

Save Reset

User-Defined AWT Resource Limits

You can define custom AWT resource limits for utilities, and TASM will either reject or delay utilities that exceed these limits. To create custom limits, in the **Throttles** view, use the **Resource Limits** tab.

You may want to create a custom AWT resource limit for utility jobs for these reasons:

- You want to enforce a lower percentage of AWTs for utility jobs than the system default.
- You want to create separate AWT resource limits for utility jobs from different users or applications.

Note:

When you create a custom AWT resource limit for an application or user, the system-level rule and the application-specific or user-specific resource limit are enforced separately. However, all applicable rules must be satisfied before a utility can run.

For example, the DBA defined the following AWT resource limits in Viewpoint Workload Designer:

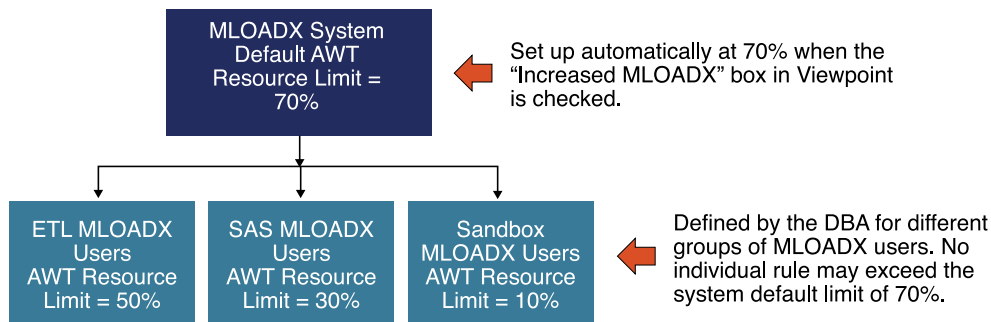
- Rule 1: A limit of 45 percent of AWTs for all DSA restore jobs in the system, with delay
- Rule 2: A limit of 15 percent of AWTs for DSA restore jobs from user Lee, with delay

With these limits, the following scenarios can occur:

- Scenario #1: Only DSA restore jobs from user Lee are running. These jobs together cannot use more than 15 percent of AWTs.
- Scenario #2: User Lee is not running any DSA restore jobs. Other users are running DSA restore jobs and consuming the limit of 45 percent of AWTs. When user Lee submits a DSA restore job, TASM delays it because it would exceed the AWT limit of Rule 1 (45 percent of AWTs for all DSA restore jobs).

You can see all the AWT Resource Limit rules in the **Resource Limits by State** tab in the **Throttles** view.

The following illustrates custom AWT resource limits for different groups of users.



Keep in mind the following AWT requirements for each utility protocol, so that you can select the appropriate limits for your environment.

Utility Protocol	AWTs Required To Start
FastLoad	3
MultiLoad	2
FastExport No Spool	2
MLOADX	MIN(2, number of target tables)
DSA Backup	2
DSA Restore	3*

* TASM initially assumes a DSA restore job needs 3 AWTs. Actual AWT usage, which can be up to 55 AWTs, will be dynamically updated later in the job and will affect the next utility job.

Utility Session Rules

Utility session rules specify the number of sessions that a utility job can use. You might want to create custom utility session rules for different request sources, such as applications, users, and profiles.

To create utility session rules, in the **Workload Designer** portlet access the **Utility Sessions** tab from the **Sessions** view. Utility session rules override the minimum and maximum session limits in scripts (the MINSESS and MAXSESS parameters). For non-conforming utilities, utility session rules override only the maximum session limits.

Data Stream Architecture (DSA) does not use utility sessions. Use the **Workload Designer** portlet to specify the number of build processes to be used for a DSA job.

When you specify the utilities to which the rule applies, you indicate the following:

- Utility name
- Number of sessions (which cannot exceed the displayed default system limit)
- Optional qualification criteria:
 - Request source
 - Query band

Default Utility Session Rules

Vantage assigns a default number of sessions to each utility based on the following factors:

- The specific utility (for example, FastLoad or MLOADX)
- The number of AMPs in the system
- Data size (any, small, medium, large)

The default data size is medium.

Tip:

Enable the Viewpoint **Monitored Systems** portlet **System Stats** data collector so that the **Workload Designer** portlet can retrieve the correct number of AMPs in the system.

User-Defined Utility Session Rules

You can customize utility session rules by doing either of the following:

- Create a custom rule with fewer sessions than the system default.

This option is useful if the client machine is not powerful enough to use all the sessions at the same time or if you want to restrict utility sessions to make more sessions available to other job types.

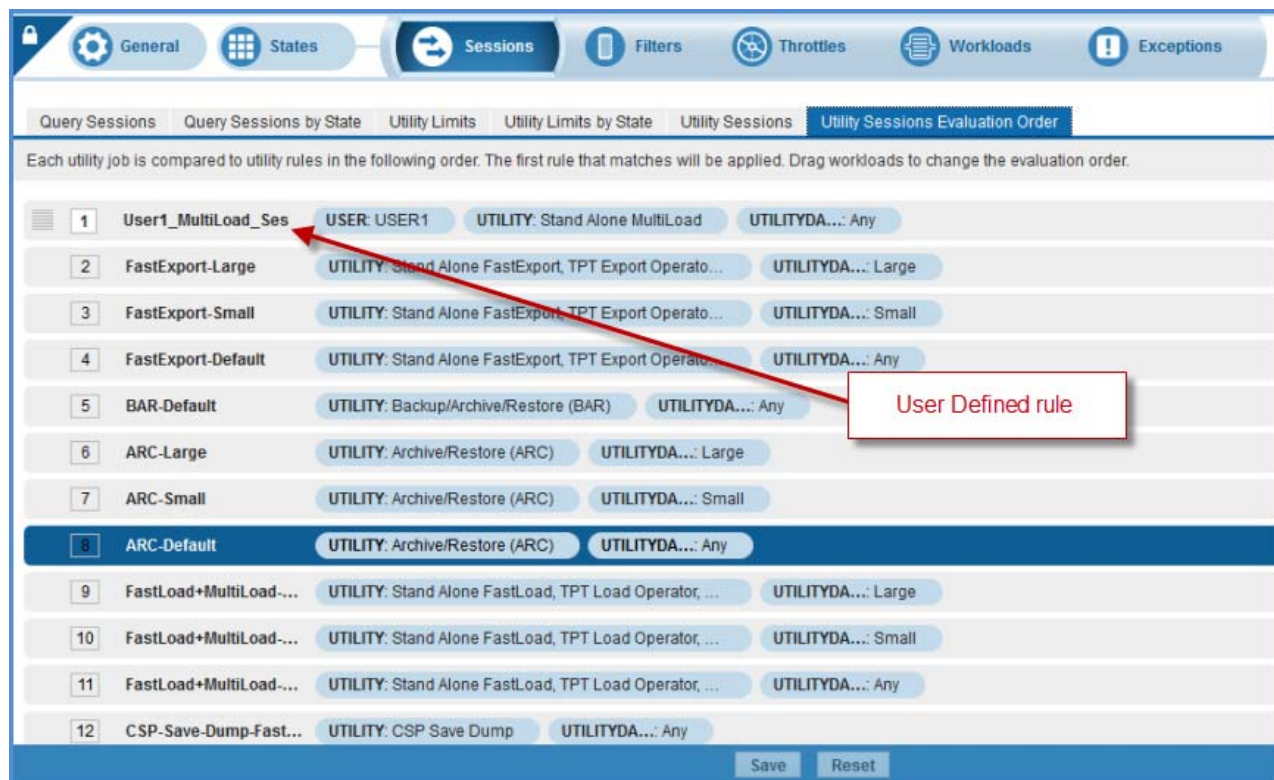
- Use the UtilityDataSize session query band to specify a small or large data size for the session, as in the following example:

```
SET QUERY_BAND='UtilityDataSize=LARGE;' UPDATE FOR SESSION;
```

For more information about UtilityDataSize, see *Teradata Vantage™ - SQL Data Definition Language Detailed Topics*, B035-1184.

Utility Session Rule Evaluation Order

TASM evaluates user-defined utility session rules before default session rules. The evaluation order for user-defined utility session rules can be changed by accessing the **Utility Sessions Evaluation Order** tab in the **Sessions** area, as follows.



Specific rules can be dragged and dropped to change the order in which they are evaluated. You cannot change the evaluation order of default rules.

Tip:

Place utility session rules with more restrictive classification criteria higher in the evaluation order. When multiple session rules could apply to a utility, TASM always uses the first matching rule.

Recommendations for Utility Session Rules

Follow these guidelines when setting up utility session rules:

- Use the default utility session rules wherever possible. Change the system-generated values if necessary. These rules should apply to most utility jobs.
- Create additional rules, if needed, for specific request sources or query bands, such as application, user, or job type.
- Adjust the evaluation order of user-defined rules.

Key Recommendations for Utilities

Teradata recommends that you manage utility workloads separately from non-utility workloads. Take the following actions to achieve this:

- Create workloads (with optional throttles) for utilities that classify by utility type.
- Create separate workloads with no throttle for all utility requests outside of utility units of work. The classification criteria of these workloads should contain only the application names of the utilities and no utility type.
- Exclude utility application names from system throttles.

These steps ensure that utilities, which have different resource requirements, are not mixed in with non-utility work.

Workload Exceptions

Exception Management

TASM monitors and manages running requests and can act if a request exceeds expected limits. Administrators can use the **Exceptions** area on the Ruleset toolbar to define behavioral limits and the remedial actions TASM should take when a limit is exceeded. Tactical workloads have system-provided exceptions.

You can create exceptions as follows using Workload Designer:

- **By Planned Environment:** Within each planned environment, specify which exceptions are enabled for a workload.
- **By Workload:** For each workload, specify which exceptions apply in a planned environment.
- **By Exception:** For each exception, specify the workloads and planned environment for which it is enabled.

Exception Criteria Options

Two types of user-specified durations may be associated with exceptions:

- **A threshold.** The specified action occurs as soon as the threshold is exceeded. Most exceptions are in this category. These include the following exceptions:
 - Maximum spool rows
 - Spool usage in bytes
 - I/O count
 - Spool size
 - Blocked time
 - Elapsed time (with the options to exclude blocked time and exclude throttle delay time)
 - Number of AMPs
 - CPU time (summed over all nodes)
 - Tactical CPU usage threshold per node
 - Tactical I/O physical bytes per node
 - I/O physical bytes
- **A threshold and a qualification period.** The specified action occurs only after the exception has persisted beyond a certain value for a specified time. The following exceptions are in this category:
 - CPU Disk Ratio
 - Skew (I/O or CPU skew difference or skew percentage)

Exceptions can have one or multiple criteria. When an exception has multiple criteria, all criteria must be exceeded to trigger the action.

Select values for exception criteria based on typical values for requests in that workload. Identify boundaries that requests in that workload should not cross. For example, if the typical request in the Finance workload takes about 250 CPU seconds, and almost all requests in the workload range from 100 to 1,000 CPU seconds, any request that exceeds 1,000 CPU seconds is out of place in the Finance workload and is running at the incorrect priority. In this case, you might create an exception to take automated action on any request that exceeds 1,000 CPU seconds.

Tactical Workload Exceptions

All Tactical workloads have an associated **Tactical Exception** that identifies and acts on requests that have non-Tactical characteristics. If a resource-intensive request starts to run in a Tactical workload, TASM demotes that request to another workload.

TASM monitors the following criteria, either of which can trigger a tactical exception:

- **Tactical CPU Time**
- **Tactical I/O Physical Byte**

TASM changes the workload of a request if the request exceeds either of the following **Tactical CPU Time** thresholds:

- **CPU (sum over all nodes)** in CPU seconds
- **CPU per Node** in CPU seconds

TASM changes the workload of a request if the request exceeds either of the following **Tactical I/O Physical Byte** thresholds:

- **I/O (sum over all nodes)** in KB, MB, or GB
- **I/O per Node** in KB, MB, or GB

When a request exceeds a per node limit, the request is demoted on that one node only. When a request exceeds a sum over all nodes limit, the request is demoted across all nodes. It is possible that a request can exceed a per node limit on one or more AMPs but not exceed the sum over all nodes limit. In that case, the request could run in different workloads on different nodes. One way to keep requests running in the same workload on all nodes is to set the per node and sum across all nodes thresholds to the same value.

If DBQLogTbl logging is enabled, the number of nodes that reach the per node threshold is logged in the following fields:

- CPU: TacticalCPUException
- I/O: TacticalIOException

By default, the sum over all nodes CPU and I/O thresholds are automatically set at the per node setting multiplied by the number of nodes in the system. To change this default, use the **Tactical Exception** tab on the **Workloads** view.

Tip:

If you add or replace hardware, TASM does not automatically adjust the CPU and I/O sum over all nodes settings. Always recheck these values after a reconfiguration to make sure they fit the new platform.

Tip:

Teradata recommends that the **Tactical CPU Time** thresholds **CPU (sum over all nodes)** and **CPU per Node** be set above 1 second. Checking more often than every 1 second can add considerable overhead and impact active work.

CPU Disk Ratio

The **CPU Disk Ratio** exception detects requests that have an unusually high ratio of CPU use versus logical I/Os. One example of this is an accidental unconstrained product join on a very large table. This metric is also called the product join indicator (PJI). Other legitimate requests, such as full table scans, can also be CPU-intensive, however.

Skew Exception Criteria

A skewed request can severely impact the parallel efficiency of the system, eliminating the benefit of multiple AMPs. If the system is processing one skewed request step, all the AMPs with less processing to do must wait until the single skewed AMP with more processing to do completes. Only the skewed AMP runs at capacity while the other AMPs sit idle. In a mixed workload, the skewed AMP cannot respond as quickly as the other AMPs because it is clogged with skewed processing. All requests in the system can slow down when there is a skew.

Detecting Skew

Either CPU or I/O skew percent exceptions can indicate skew. TASM exceptions detect skewed requests based on the number of AMPs active in the request step. A group-AMP request that uses about the same processing time for each AMP involved would not be detected as skewed, even though some AMPs are not affected by the request.

Method for Detecting Skew

In general, TASM detects skew successfully using just the skew percentage metrics. However, some situations can lead to a false detection of skew. In a multi-step request, one step could be very skewed but also short and not indicative of an overall problem with the request. It is important with skew detection that temporary skew within a request not be interpreted as persistent skew and that temporary skew be ignored.

Tip:

Do not set the qualification time too low on skew exceptions. The skew condition should persist long enough to qualify as a problem with the request.

Synchronous and Asynchronous Exception Checking

TASM does not check for skew synchronously at the end of request steps. Instead, TASM detects skew asynchronously at the end of the **Exception Interval** period specified in the **Other** tab on the **General** view. TASM does not wait until the end of a request step to detect skew because a long skewed step could overtake system resources.

Exception Actions

When you create an exception, you define what TASM should do automatically if it finds the exception.

Choose from the following actions.

Action	Description
Notification Only	Sends notification and takes no other action.
Abort	Stops the query.
Abort Selects Only	Stops the query only if it is a SELECT.
Change Workload to	Changes the query to the workload you select from the list.

For all exceptions, TASM logs the exception to TDWMExceptionLog and captures the SQL in DBC.DBQLSqlTbl.

Note:

A **Change Workload to** action will be overridden and ignored if the PM/API request TDWM WD ASSIGNMENT has been issued and it affects this request.

Notifications

You can select one or more of the following **Notifications** to occur when an exception is detected.

Notification	Description
Send Alert	Specifies the action to trigger.
Run Program	Specifies the Alerts registered programs to trigger.
Post to QTable	The string you enter is posted to the QTable.

Note:

To use alerts as a notification option, you must first create the alerts in the **Alert Setup** portlet.

Using Alerts or Run Program with Exceptions

Before you can choose the **Send Alert** or **Run Program** actions, you must define those actions in the **Alert Setup** portlet. For more information, see [Using Alerts or Run Program](#).

Use the **Alert Viewer** portlet to see the alerts. Alerts include useful details about the request that triggered the exception, such as the host ID, session number, and request number.

Using Post to QTable with Exceptions

The **Post to QTable** exception action posts a notification to the system notification queue (DBC.SystemQTbl). For more information, see [Using Post to QTable](#).

Elapsed Time or Blocked Time Considerations

The **Change Workload to:** action is not available for the **Elapsed Time** or **Blocked Time** exception criteria. The purpose of changing a workload is to put a request where it belongs in the priority hierarchy. Temporary system-wide conditions cause a request to be blocked or run too long. These exceptions do not occur because the request is out of character for the workload.

Elapsed Time or **Blocked Time** exceptions are better suited to an automated alert. Then the DBA can investigate the system-wide situation and resolve the problem by identifying the cause of the slowdown, rather than punishing the requests suffering from the situation.

Abort Exception Action Types

Consider the possible implications of an abort before deciding to use that exception action. The following examples show why an abort could cause issues.

Request Description	Effect of an Abort
A full-table update that has been running for 30 minutes and has only 5 minutes remaining.	A rollback would take several times longer to undo than completing the request. Consider instead the Abort on Select option.
Part of a multistatement request in which the results of one request feed into the next request.	Aborting one of the requests could result in inaccurate results.
It is considered an exception because of skew.	The wrong request might be aborted due to a false skew detection.

Even if you know that the workload does not receive the type of request specified in the exception, be careful. TASM may abort a request that is important to the business.

Exception Action Conflict Resolution

One request could trigger multiple exceptions. If that happens, TASM performs all exception actions that do not conflict. For example, TASM always executes all **Send Alert**, **Run Program**, and **Post to QTable** actions because they do not conflict. TASM always performs the associated logging also.

A conflict occurs when two simultaneous exceptions call for the following actions against a request:

- **Abort** versus **Change Workload**
- **Change Workload to: Workload A** versus **Change Workload to: Workload B**

TASM follows these rules to resolve exception action conflicts:

- Abort actions take precedence over **Change Workload to:** actions
- **Change Workload to:** is applied based on the workload management method the workload uses, in ascending order of precedence:
 - Timeshare tier
 - SLG tier
 - Tactical tier

TASM resolves further exception action conflicts with these rules, which favor the lower workload classification:

Conflict	TASM Rule
Two Change Workload to: actions move a request to the Timeshare tier but to workloads on different levels.	The request moves into the workload on the lower level. If both actions move the request to the same level, the request moves into the workload whose name appears first in the alphabet.
Two Change Workload to: actions move a request to the SLG tier but to workloads on different levels in that tier.	The request moves into the workload on the lower level. If both actions move the request to the same level, the request moves into the workload with the lower workload allocation percentage value. If the two workloads have the same allocation percentage value, the request moves into the workload whose name appears first in the alphabet.

In every case, TASM logs the exception actions that it did not perform into TdwmExceptionLog and indicates that they were overridden.

Apply Change WD Action Consistently

TASM can enable or disable exceptions for different planned environments. For example, at night TASM might abort a request matching exception criteria, but during the day, TASM might let that request run.

Tip:

Teradata recommends that **Change Workload to:** actions name the same workload across all planned environments.

Key Recommendations for Workload Exceptions

Send notifications if requests cause one of the following exceptions:

- **CPU Disk Ratio** (this is an indicator of a product join)
- Skew

The request may need to be fine-tuned. Consider aborting the request if the situation persists. You may decide to turn on an exception notification to assess the impact before setting an abort action.

Filters

System Filters

Filters are TASM rules that reject user logons or reject requests before they execute. For example, you may want to prevent requests from executing against the Pricing table every Friday afternoon while it is being updated. After defining a filter, you can apply the filter to specific states.

Tip:

Teradata recommends that DBAs use filters with caution and apply them selectively. It may be inappropriate to reject requests in an ad hoc environment.

Examples: System Filters

The following examples show cases in which system filters are useful.

Problem	Reason for Filtering out the Request
A request accesses a large, detailed table using an unconstrained product join.	A request with no join constraints against a large table is poorly written and uses too many resources. A DBA will have to abort this type of request anyway. Rejecting the request before it runs prevents resources from being wasted on a request that will never finish.
A request uses a full-table scan on very large, detailed data tables, such as call detail data or deep history data, and on Native Object Store tables. Requests that use indexed access would still be allowed to run.	Full-table scans on very large tables could prohibitively impact other users, particularly at peak processing times. If such access is appropriate for a small group of users, a DBA can create a filter that excludes those users. If this type of filter were active only during normal daytime hours, batch reports or maintenance jobs could execute without issues during off-hours.
Maintenance activity requires that all users log off the system, while requests in progress be allowed to complete.	New requests must be prevented from running. After the maintenance work is done, the DBA can disable the filter.

The preceding examples show system-wide filters, in which all requests are candidates. However, you could simulate a filter at the workload level. For example, if you want to reject all queries submitted to WD-B while in the CriticalFocus state, set a workload throttle using the “Reject” option.

Warning Mode

When you create a filter, you can choose **Warning Only** mode for initial implementation. This mode returns a warning message when the filter affects a request instead of rejecting the request, so you can

determine how the filter is functioning without disrupting users. Requests that match filter criteria are logged in DBQLogTbl, with the “WarningOnly” field set to “T.” Once you determine that the filter is working as planned, you can turn off **Warning Only** mode and allow TASM to reject affected queries.

Tip:

Teradata recommends that you place a new filter in **Warning Only** mode for 1-2 weeks so that you can predict how many requests the filter will affect when it goes live. The **Warning Only** period also gives you time to work with users who may not realize that they are submitting poorly performing requests.

Working with Classification Criteria for Filters

Filters use the same classification criteria as workloads. For more information, see [Classification Criteria](#). Multiple criteria of the same type (such as multiple users) are ORed together. Multiple criteria of different types (such as a table and an account), are ANDed together.

Note:

All request target objects are considered to be of the same type and are ORed together.

State-Specific Settings

When you create a filter, you can enable or disable it for use in predefined system states. For example, you can enable a filter that rejects long-running requests when the system is in a degraded state. After the system is in **Base** state again, TASM turns off the filter.

Key Recommendations for Filters

Use filters carefully after an evaluation period in **Warning Only** mode.

Arrival Rate Meters

Arrival Rate Meters

Arrival Rate Meters (ARMs) are TASM rules that provide a way to regulate the flow of requests admitted by TASM into the system. An ARM rule specifies the maximum number of requests per time unit that TASM will admit into the system. Qualification criteria can be used to target specific types of requests. For example, three requests per hour for requests with estimated processing time of more than one hour.

The TDWM order of processing is:

1. Filters
2. ARMs
3. Throttles

Examples: Arrival Rate Meters

The following examples show cases in which ARMs are useful:

Problem	Solution
Flood of requests from a specific application monopolizes the front of the throttle delay queue. Subsequent requests end up at the end of the delay queue.	Create an ARM rule for this application so that other requests can get in between each time unit and can be serviced earlier.
Complex requests use too much resource.	Create an ARM rule for complex requests to allow only a few requests each hour during weekdays and a higher limit on weekends.
Certain applications can be extremely resource intensive because they generate a very large number of table scans.	Create an ARM rule for table scan requests from the application to spread out the requests over time to reduce peak resource usage by the application.

Working with Arrival Rate Meters

An ARM is a time-based flow rate rule. These are ARM rule attributes:

Attributes	Description
Time Unit	Time period for request rate (second, minute, hour).
Limit	Maximum request rate per time unit (3 requests per minute).

Attributes	Description
Qualify Time	Number of seconds that the current ARM rate must continuously remain at or above the limit before action is taken. A zero value means action is immediately taken as soon as the limit is reached.
Qualification	Common classification criteria (Request Source, Query Characteristics, Target, and Query Band) can be specified to target specific types of requests (user, application, table scans, processing time, and so on).
Action	Rule action performed when Limit and Qualify Time conditions are met. Actions include Delay, Reject, and Logging (warning mode). Limit, Reject, and Delay action attributes are state-dependent values.

Logging (warning mode) is similar to that of Filters. See [Warning Mode](#).

For more information, see *Teradata Vantage™ - Application Programming Reference*, B035-1090 and Teradata Viewpoint Workload Designer in *Teradata® Viewpoint User Guide*, B035-2206.

Usage Notes

- Only the Collective rule type is supported for an ARM rule. Individual and Member rule types are not supported.
- Each ARM rule is limited to a single time unit. Time units cannot be mixed across states. If a secondary state has a different time unit than the base state, the time unit for the secondary state is ignored and the time unit for the base state is used instead. A warning will be written to the DBC.TDWMEventLog table.
- TDWM tracks the number of admitted requests within the given time unit for each ARM rule.
- Each ARM counter is divided into 6 equal intervals for better responsiveness. For example, a minute based ARM rule is divided into six 10-second intervals.
- The current arrival rate is the sum of the 6 interval counts.
- Each interval has a start time, an expiration time, and a query arrival count. An example of a minute based ARM rule:

Interval Number	Start Time	Expiration Time	Number of Arriving Requests
1	8:00:00	8:01:00	1
2	8:00:10	8:01:10	2
3	8:00:20	8:01:20	0
4	8:00:30	8:01:30	4
5	8:00:40	8:01:40	2
6	8:00:50	8:01:50	1
Current Arrival Rate = 10/minute			

- When an interval expires, a new interval is created with a new start time, expiration time, and interval count. The expired interval count is logically subtracted from the ARM count. If the ARM count is below the limit, TDWM checks to see if any deferred queries can be admitted into the system. TDWM can keep admitting deferred queries until the ARM limit is reached.
- The ARM count is based on query admission time. The elapsed time of the query itself is not relevant. It is possible for the ARM count to drop to zero but still have associated queries running (for example, a second based ARM rule where queries run for several seconds). Conversely, it is possible for queries to complete but still be counted against the ARM rule (for example, an hour based ARM rule where queries run for several seconds).
- If an ARM rule is defined with a non-zero qualify time, it is possible for the current ARM rate to exceed the limit until the required qualify time is met.

This is an example of ARM counter behavior with an ARM limit of 10 requests per minute with 0 qualify time:

Interval Number	Interval Time	Expiration Time	Number of Arriving Requests	Current Arrival Rate	Defer Queue Length	Comment
1	8:00:00 - 8:00:09	08:01:00	1	1	0	First request is admitted. Current rate is set to 1.
2	8:00:10 - 8:00:19	08:01:10	2	3	0	Second and third requests are admitted. Current rate is set to 3.
3	8:00:20 - 8:00:29	08:01:20	2	5	0	Fourth and fifth requests are admitted.
4	8:00:30 - 8:00:39	08:01:30	3	8	0	Sixth, seventh, and eighth requests are admitted.
5	8:00:40 - 8:00:49	08:01:40	2	10	0	Ninth and tenth requests are admitted.
6	8:00:50 - 8:00:59	08:01:50	1	10	1	Eleventh request is added to Defer Queue because current arrival rate has reached the limit.
7	8:01:00 - 8:01:09	8:02:00	3	10=>9=>10	3	The first request is no longer counted in the current rate. Current rate drops to 9. One deferred request (for

Interval Number	Interval Time	Expiration Time	Number of Arriving Requests	Current Arrival Rate	Defer Queue Length	Comment
						example, eleventh) is released. Current rate goes back to 10. The 3 new requests arriving in this interval are added to Defer Queue (total of 3).
8	8:01:10 - 8:01:19	8:02:10		10=>8=>10	1	The second and third requests are no longer counted in the current rate. Rate drops to 8. Two deferred requests are released. Current rate goes back to 10. No new request arrives in this interval. One request remains in Defer Queue.
9	8:01:20 - 8:01:29	8:02:20		10=>8=>9		The fourth and fifth requests are no longer counted in the current rate. Rate drops to 8. One deferred request is released. Current rate goes up to 9. Defer Queue is empty.
10	8:01:30 - 8:01:39	8:02:30		6		The sixth, seventh, and eighth requests are no longer counted in the current rate. Rate drops to 6.

Key Recommendations for Arrival Rate Meters

Use arrival rate meters carefully after an evaluation period in warning mode.

States and Events

States and Events Overview

The best practice under normal operating conditions is to use only one ruleset. Instead of creating multiple rulesets for different situations, you can create states to define how TASM allocates resources during different times of the day, different days of the month, or when system performance is degraded. Moving from one state to another, due to planned or unplanned circumstances, results in an automatic change in the workload management settings. Only one state in one ruleset can be in effect at a time. State definitions consist of these factors:

- Health conditions: These define system health, for example, a degraded health condition. You can define an unplanned event that triggers a health condition, such as a down node.
- Planned environments: These are time frames or different processing windows that the system supports, for example, a Late-Night planned environment for batch jobs.

Administrators define events that cause a health condition or a planned environment to change. A change to either or both causes a state change, depending on how the state matrix is defined. Events include the event definition and the actions that Vantage takes when the event occurs. The event definition can be based on the occurrence of a single event or a combination of events. When an event occurs, Vantage can take one of these actions:

- Change the health condition or planned environment
- Send an alert
- Run a program
- Post a notification to the system queue table (DBC.SystemQTbl)

Once you create a state, you can apply filters, throttles, and workloads based on that state.

State Matrix Basics

TASM comes with a default state matrix. You will probably want to expand this for your environment.

A state matrix shows the intersection between planned environments and health conditions. The intersection of these two categories is a state.

The two state matrix categories are as follows:

- Health conditions: This shows system health. When a node is down, you might want more restrictive rules. Health conditions appear on the vertical axis of the matrix. You may want to define two or three different health conditions. This is the "what" category.
- Planned environments: This shows the different processing windows that a system is expected to accommodate, such as, night and day, weekday and weekend, month-end, or year-end. Planned environments are on the horizontal axis of the state matrix. This is the "when" category.

Note:

TIWM includes planned environments only.

The state matrix lets you enforce different rules in different circumstances. Multiple planned environment and health condition pairs can be associated with the same state.

First, use Viewpoint **Workload Designer** to define different states. Then drag and drop each state into the location in the state matrix where you want it to be active. The same state can be used many times. The following figure shows a simple but effective state matrix.



The default state matrix is 1 x 1. A planned environment called **Always** is defined for 24 hours x 365 days a year, and a health condition called **Normal** is perpetually in effect. This combination has a default state called **Base**. If there are multiple planned environments and health conditions, each unique pair can have a unique state. Each state can have unique throttle limits and workload attributes. Only one state is active at a time, based on the highest-severity health condition and highest-precedent planned environment in effect.

When TASM evaluates the planned environments to see which one should be active, it searches from the right-most state (the one with the highest precedence) and moves left, stopping at the first planned environment that fits the criteria. For health conditions, the search starts at the bottom, with the one with the highest severity, moves up, and stops at the first health condition that qualifies. That way, if more than one of those entities is a candidate for being active, TASM selects the one that is most important or most critical first.

Working Values

Some parameters in rulesets can be changed in different states and some cannot. You can customize system-level attributes, such as filters and throttles, for different states. Different states can have filters and throttles enabled or disabled or they can have different throttles limits.

You can also customize some workload characteristics for different states. To understand what you can change about a workload when a state change occurs and what you cannot change, consider that workloads contain these two types of properties:

- Fixed attributes: The details that define the workload, which include the following items:
 - Classification criteria
 - Exception definitions and actions
 - Position (or tier) in the priority hierarchy
 - Evaluation order of the workload
- Working values: The variables that are part of the Workload Definition, which include the following items:
 - Workload or virtual partition share percent values
 - Workload throttles
 - Exception enabling or disabling
 - Service Level Goals
 - Minimum response time (called **Hold Query Responses** in Viewpoint **Workload Designer**)

The following workload working values may vary based on the planned environment in effect:

- Service Level Goals
- **Hold Query Responses**
- Exception enabling or disabling
- Workload management method priority values (workload distribution or timeshare access levels)

The following settings may vary based on the state in effect:

- Session controls
- Workload throttles
- System throttles
- Resource limits
- Filters
- Query session limits
- Utility limits

Often, workloads do not keep the same level of importance throughout the day, week, month, or year. A load workload may be more important at night, and a request workload may be more important during the day. Also, when the system is degraded, it may be more important to complete tactical workloads than strategic workloads.

State Matrix Considerations

You rarely need more than a few states. A simple matrix is easier to set up, monitor, and tune. Keep the size of the state matrix and the number of unique states to a minimum.

If there is no need for additional states, such as when you start using TASM, Teradata recommends that you use the default **Base** state. While it is easy for TASM to change states, workload management is simpler when there are fewer states to consider. Follow these guidelines to manage states well:

- Add more states if needed, but keep the number to a minimum. You want to keep the planned environment-related state transitions down to two or three per day.
- Do not create a unique health condition and a related state for each possible way system health could be degraded. Instead, consider creating one or two new system conditions to represent all possible system degradation scenarios.

Health Condition Duration

When you define a health condition, you can associate it with a minimum qualifying duration. To understand why, consider the following scenario. The system has a RED health condition for degraded health and a GREEN health condition for good health. When an event causes the RED health condition, the state associated with RED goes into effect. RED may have lower throttles limits, more filters, and a lower workload share percent. If, after it transitions to RED, the system immediately returns to good health, the system could transition to the GREEN health condition and the more restrictive values could be lifted.

To prevent continual state transitions, set a minimum qualifying duration for health conditions. This gives the system a better chance of resolving the issues that are causing the degraded state. Teradata recommends that health conditions that are caused by internal events (not user-defined events) be set to a minimum duration of 10 minutes (600 seconds).

Events

All system activities are called events. TASM detects when they occur and acts based on the rules you define for that event. The following are the two types of events:

- Planned environment events, such as a batch window or a scheduled time for month-end processing
- Unplanned events, such as a down AMP, that trigger a system health condition

You can define your own planned environment and system health events. You can combine one or more planned events into a planned environment. To define events in Viewpoint **Workload Designer**, edit the state matrix directly or select the **Setup Wizard** button in the **States** view.

TASM evaluates and acts on events based on the **Event Interval** you define in the **Other** tab of the **General** view in Viewpoint **Workload Designer**.

Event Types

The following table shows the types of events that TASM can detect either individually or in combination.

Class	Subclass	Description
Planned Events	Period	Event Frequency and Duration can include these user-specified time periods: <ul style="list-style-type: none"> • Day of Week • Day of Month and Month of Year

Class	Subclass	Description
Note: You can combine multiple events. Event combinations can trigger the same actions as individual events.		TASM monitors the system time and automatically triggers an event when the period starts. The event persists until the period ends.
	User-defined	These events can be activated at any time through an API. They last until the DBA rescinds them or until they time out.
Unplanned Events Note: You can combine multiple events. Event combinations can trigger the same actions as individual events.	System	System Events can be triggered by any of the following criteria: <ul style="list-style-type: none"> • Node Down (user-specified percent of clique) • AMP Fatal (limit) • Available AWTs (number in user-specified qualification period) • Gateway Fatal (limit) • PE Fatal (limit) • Flow Control (number of AMPs in user-specified qualification period) • I/O Usage (percent in user-specified qualification period) • CPU Utilization (percent in user-specified qualification period) • CPU Skew (percent in user-specified qualification period) When TASM detects this type of event, it keeps the event active until the component is restored or the resource value returns to normal for a predetermined time.
	User-defined	These events can be activated at any time through an API. They last until the DBA rescinds them or until they time out.
	Workload	Workload Events can be triggered by any of the following user-specified criteria: <ul style="list-style-type: none"> • Active requests (number per user-specified qualification period) • Arrivals (of requests) at a user-specified rate • AWT Wait Time (number of user-specified seconds per qualification period) • CPU Utilization (as a percentage per qualification period) • Delay Queue Depth (number of queries in the delay queue) • Delay Queue Time (in hours, minutes, or seconds)

Event Combination Events

You can combine multiple events. Event combinations can trigger the same actions as individual events.

Period Events

The period event is only available as a planned environment. Define period events to be contiguous, as follows:

- Daytime: 8:00 a.m. to 5:00 p.m.
- Nighttime: 5:00 p.m. to 12:00 a.m.

And not:

- Daytime: 8:00 a.m. to 4:59 p.m.
- Nighttime: 5:00 p.m. to 11:50 p.m.

In the latter case, at 4:59 p.m., TASM would change the state to something not associated with Daytime or Nighttime (perhaps the default **Always**). Then at 5:00 p.m., the state would change again to the one associated with Nighttime.

Wrap Around Midnight Option

When you create period events, use the **Wrap Around Midnight** option to define a time range spanning midnight. With this option, the period begins on the days selected and ends the following day. If you do not use this option, the period is only active on the days selected. The period may start or end at midnight, depending on days selected.

For example, in the following tables, you want to specify that an after-hours batch window for payroll updates occurs on Mondays and Tuesdays from 5 PM until 8 AM. In the following table, the **Wrap Around Midnight** option is not selected. The unintended consequences of not using **Wrap Around Midnight** appear in *italics*.

Time Segment	Monday	Tuesday	Wednesday
Midnight — 8 AM	<i>Active</i>	Active	<i>Inactive</i>
8 AM — 5 PM	Inactive	Inactive	Inactive
5 PM — 11:59 PM	Active	Active	Inactive

In the following table, the **Wrap Around Midnight** option is selected, and the after-hours batch window for payroll updates occurs on Mondays and Tuesdays from 5 PM until 8 AM, as expected.

Time Segment	Monday	Tuesday	Wednesday
Midnight — 8 AM	Inactive	Active	Active
8 AM — 5 PM	Inactive	Inactive	Inactive
5 PM — 11:59 PM	Active	Active	Inactive

System Events

A system event is an unplanned event, such as a down node. Incorporating system events into health conditions in the state matrix gives you greater control over the actions Vantage takes when the unexpected happens. To create an event that only sends out a notification, create the event, but do not assign it to a health condition. When the event occurs, TASM performs the notification you specify.

To define a system event, edit the state matrix directly or, in the **States** view, add to **Available Events** on the **Define and Map Unplanned Events** step in the **Setup Wizard**.

Node Down

When you create a node down event, you specify the maximum percentage of nodes that can be down in a clique before the event triggers.

When a node goes down, its vprocs migrate, increasing the amount of work on active nodes. This extra work slows performance. In that degraded situation, you may want to throttle lower-priority requests or enable filters to ensure that critical requests can still meet their SLGs. You may also want to send a notification so that you can take follow-up actions.

Recommendation: Your system may be designed to run with some degradation (a large system with hundreds of nodes may be sized expecting that there is always one node down somewhere). In that case, Teradata recommends that you set the threshold so that a node down event occurs only when the degradation exceeds what the system can absorb.

If your system is not sized to expect down nodes (as is the case with many small-to-medium-sized systems): a good threshold for a down node event is the default, 24%.

AMP, Gateway, and PE Fatal

You can define specific events for AMP, PE, and Gateway vprocs. These events detect a vproc as being fatal at system startup only. These events are similar to node down events, except you define the number of fatal vprocs that trigger the event.

Available AWTs

AMP Worker Tasks (AWTs) are the tasks inside each AMP that do database work. This work may relate to user requests or internal software functions, such as deadlock detection, error logging, or aborts.

Each AMP is given a number of AWTs at startup. If all AWTs are busy when new work arrives, the work waits in a queue until an AWT is free.

You can create a system event based on **Available AWTs**. To create the event, you specify the **# of AMPs** that have the number of **Available AWTs** you specify. For example, you can specify that the **Available AWTs** system event triggers if 2 AMPs each have only 1 available AWT. There are two ways

to interpret **Available AWTs**: AWTs reserved only for new work or the entire unreserved pool. You can define **Available AWTs** appropriately for your site in the **Other** tab of the **General** view. In most systems, there is a limit of 50 AWTs that are available to support new work.

Note that if this event has a qualification time, the threshold does not have to be maintained on the same AMP over the entire qualification period. This means that if you define a threshold of 2, and AMP A crosses this threshold on one event sample and AMP B crosses the threshold on the next event sample, the event condition is considered to be maintained across the two samples.

For more details on AWTs and flow control, see the Orange Book, *Teradata AMP Worker Tasks and ResUsage Monitoring*, 541-0009643.

AWT Usage Considerations

It is normal during busy times to have some AMPs run out of AWTs. This condition often lasts only a second or less. However, system response times degrade if there is a persistent AWT shortage.

If TASM detects an AWT shortage, analyze your corresponding CPU and I/O utilization metrics. Most Vantage systems reach 100% CPU utilization while there are still many AWTs available. Some sites experience peak throughput when as few as 40 AWTs are used. Most likely, the bottleneck is not AWTs, but CPU or I/O resources.

However, an ongoing AWT shortage may indicate an overly busy system. Remember that response time is a function of throughput capabilities and concurrency levels. If a request competes for system resources with 50 other requests at the same priority, it cannot be expected to respond as fast as when it executes alone on the system. Because AWT shortages are a rough gauge of request concurrency levels, you can use them to notify applications or users to expect slow response times. This can sometimes result in decreased user demand because some users postpone issuing their next requests until the system is quieter.

Flow Control

Flow control means that some tasks cannot send more work to busy AMPs. If many AMPs are in flow control, response times degrade and become inconsistent. This is a more serious situation than simple AWT depletions because request priorities are not honored while messages are being retried.

To define a system **Flow Control** event, specify the number of AMPs that report flow control during the **Qualification** period. TASM does not distinguish between being in flow control for 1 msec and being in flow control for the entire interval. Different AMPs can be in flow control during the **Qualification** period, as long as the **# of AMPs** criterion is met.

CPU Utilization

The system event **CPU Utilization** is based on the system-wide average of node CPU percent busy on the TASM/TIWM event interval. **CPU Utilization** indicates whether the system can do more work or if it is at peak capacity.

TASM uses the **CPU Utilization** event to act when the system is notably idle or busy for a qualifying period. See the following examples:

- If sustained system CPU > 95% for 15 minutes: Set the health condition to Yellow, which reduces lower-priority workload throttle limits.
- If sustained CPU < 50% (low system CPU utilization) for 20 minutes: Send a **Post to QTable**:notification to DBC.SystemQTbl to tell applications to execute background-type work.

Teradata recommends using the **Averaging** qualification to distinguish between temporary and persistent utilization. Averaging smooths out the peaks and valleys of CPU utilization patterns typical during a week.

CPU Skew

A TASM exception can detect when a request is skewed. However, you need to create a system event to detect a system-wide skew, such as an application running on one node. The system event **CPU Skew** can detect any type of skew, including session and application imbalances. Typically, when TASM detects a system skew, it sends an alert to the DBA, who investigates and acts manually.

Teradata recommends that you use the **Averaging** qualification option to distinguish between a temporary skew and a persistent skew.

If you are using **CPU Skew** on a coexistence system, adjust the triggering threshold to a value appropriate for the built-in system imbalance. For example, suppose in a perfectly balanced workload environment, the typical utilization of 10 old nodes is 95% when 10 new nodes are exhausted at 100%. Here, the built-in system skew is $(100 - 95) / 100 = 5\%$. In this case, set the **CPU Skew Skew Percent** to a value greater than 5%.

I/O Usage

The **I/O Usage** system event measures potential versus used I/O to find I/O bandwidth bottlenecks and determine their scope. The event monitors the LUNs that TASM determines are most vulnerable to throughput issues and reacts as their I/O usage nears capacity. For example:

- Monitor 10% of targeted LUNs.
- Trigger the event if 1% of those LUNs are at 80% of their potential bandwidth.
- Average the data over 10 minutes.
- Send an alert email to the DBA.

A DBA can define multiple **I/O Usage** events to monitor different bandwidth thresholds. The event is based on the TASM event interval.

Guidelines

- Teradata recommends that customers begin with the default values of monitoring 10% of targeted LUNs and triggering when 1% of monitored LUNs exceed the usage threshold. Customers can change these values later as needed.

- Due to the type of data involved, only the **Averaging** qualification method is available. For more information on qualification methods, see [Event Qualification](#).
- To increase the precision of detecting I/O bandwidth issues, increase the percentage of LUNs monitored (up to 50 LUNs) and increase the percentage of targeted LUNs to trigger from, up to 100%.
- To reduce how often the event triggers, increase the percentage of LUNs that triggers the event.

Considerations

- TDWM automatically adjusts I/O values for systems with Workload Management Capacity on Demand (WM COD). For example, if a system has 75% WM COD (uses $\frac{3}{4}$ of the system) and bandwidth is exhausted, the **I/O Usage** event reports usage at 100%. The system is using 100% of the I/O available to it.
- TDWM does not adjust I/O values for systems with Platform Metering Capacity on Demand (PM COD). DBAs need to adjust bandwidth percentages to reflect the PM COD value. For example, if the PM COD value is 80%, the site needs to react to that 80% threshold as if it were 100%.

User-Defined Events

You can create your own events and use the TDWMEventControl Workload Management API to set the event to Active or Inactive. As part of this API, you can define a how long an event is active. The event automatically becomes inactive when the time expires. Otherwise, an explicit API call is required to change the condition of the event.

To create a user-defined event, edit the state matrix directly or add to **Available Events** on the **Define and Map Unplanned Events** step in the **Setup Wizard** on the **States** view.

To create a health condition, edit **Unplanned Events**, then select **Available Events**, and then select **User Defined Event** in the **Create Event** window.

Workload Events

A workload event is something unplanned in a workload, such as too many active requests. You may want to be notified of the event without triggering other actions. However, you can use a workload event as a trigger for changing to a different health condition, which can trigger a state change. For example, select **Active Requests** as the event type and set a limit for the number of requests that can be active at one time. When the limit is reached, the workload event triggers a change in the health condition.

To create a workload event, edit the state matrix directly or add to **Available Events** on the **Define and Map Unplanned Events** step in the **Setup Wizard** on the **States** view.

Teradata recommends that DBAs create workload events only for critical workloads.

To create an event that only sends out a notification, create the event but do not assign it to a health condition. When the event occurs, it triggers TASM to send the notification you specify.

Note:

When a workload has defined events, it cannot be deleted until the events are deleted.

Active Requests

The **Active Requests** event lets you monitor how many requests are running in a workload. **Active Requests** does not include requests held in the delay queue.

If the number of active requests stays high, that indicates either unmanaged arrival rate surges and lulls or other unusual situations. Too many active requests can cause the following problems:

- The exhaustion of critical shared resources such as AWTs, memory, and physical spool
- The possibility of entering flow control or congestion management

One valuable use of the **Active Requests** event is to detect when too many requests are running in a penalty-box workload. Typically, a request enters a penalty-box workload because it is running too long and triggers an exception. Requests that are already running cannot be throttled, so you cannot use throttles to control requests in the penalty-box workload. If a request in the penalty-box workload holds any critical, shared resources, it will probably hold those resources for a long time because of the low priority given to the penalty-box. This means that when the penalty-box has many active requests, there is a strong chance that higher-priority work is affected. If more than 3 or 4 requests are running in the penalty-box workload, the **Active Requests** event action can notify the DBA so that he can abort some requests manually.

Arrivals

The workload event **Arrivals** tracks how many SQL requests are classified into a workload. You specify the time period: hour, minute, or second, and how many requests arriving in that time period trigger an event. **Arrivals** tracks arrivals only, not what happens to the request afterward, including throttling or demotion to another workload.

Arrivals can detect surges and lulls. Use this event to detect the following situations:

- Higher than usual request volumes. In a disaster, there can be more telephone calls and related requests. The DBA can use a change health condition action to adjust workload management rules automatically so that critical work can complete.
- A lull in request arrivals. A lull over the weekend should trigger an event with notification so someone can investigate. This prevents a weekend backlog of requests from turning into a flood of work on Monday morning.

Arrival rates can vary greatly. That is why Teradata recommends that DBAs use the **Averaging** qualification method for **Arrivals** and specify an appropriately long qualifying period. For example, some workloads may receive requests in bursts. If the bursts occur every 10 minutes, a DBA should use an **Averaging** interval that is a factor of 10 minutes (for example, 10, 20, or 30 minutes).

AWT Wait Time

The **AWT Wait Time** workload event detects if a workload is delayed when it tries to use an AWT. Different automated actions are appropriate for this event, depending on the workload and the length of the delay. Longer delays are okay for lower-priority work, but almost no delay is acceptable for Tactical work. For example, consider the following events and actions:

- If a Tactical workload is delayed by more than 200 msec
 - Change the health condition, which results in a state that tightens throttles on non-critical work.
- Detect when **AWT Wait Time** exceeds 5 seconds for high-priority workloads, 20 seconds for normal-priority workloads, and 5 minutes for low-priority workloads.
 - Change the health condition or just alert the DBA, depending on the workload priority.

CPU Utilization

The workload event **CPU Utilization** helps a DBA identify and solve issues before they worsen. The **CPU Utilization** event action can notify a DBA to solve the underlying issue before it causes unacceptable performance.

The **CPU Utilization** event can detect and resolve the following situations:

- A workload has taken over the system: If workload Godzilla has **CPU Utilization** > 90%, change the health condition to **Blue**, which lowers the throttle limit on workload Godzilla. Send the DBA an alert so he can investigate further.
- A workload is starved: Workload Hungry has an event combination for **CPU Utilization** and **Arrivals**. If **CPU Utilization** < 2% and Hungry has **Arrivals** > 100 per hour, send an alert to the DBA to investigate why Hungry is being starved.
- As an alternative to a user-defined event to activate a planned environment: If Night_Owl workload has CPU > 0% for 10 minutes, set the health condition to **Loading**, which lowers the throttle limits on non-load requests.
- High workload CPU utilization: If workload Popular has CPU > 50% for 30 minutes, write an event notification to DBC.SystemQTbl to notify Popular application users not to submit new work. It is assumed an application outside TASM/TIWM monitors DBC.SystemQTbl rows for this condition and then sends user notifications once this condition is detected.

Teradata recommends using the **Averaging** qualification option for this event to distinguish between a temporary **CPU Utilization** level and a more chronic one.

Delay Queue Depth and Delay Queue Time

Two workload events can detect when a workload is delayed for a long time because of throttle limits:

- **Delay Queue Time:** This event detects how long in seconds, minutes, or hours the longest-waiting request has been in the delay queue. **Delay Queue Time** can, optionally, include the time a system throttle delays a request.
- **Delay Queue Depth:** This event detects how many requests are currently delayed.

An interesting use case follows: Some applications indicate performance levels to users before they submit their requests. This levels system load because users may defer their next request when there are long response times. But even if the users submit their requests anyway, they are more satisfied because they know what to expect. In fact, one customer saw fewer users killing and resubmitting their requests when the customer established user expectations with a response time indicator. Using the response time indicator improved overall system performance because it eliminated the kill and resubmit resource consumption. You can use delay queue events to serve as an application performance indicator when you select the **Post to QTable** action. This action writes rows to DBC.SystemQTbl. Monitoring applications consume the rows in this table to provide the performance level indicators, as described here.

SLG Response Time

DBAs use SLGs to measure how well a workload performs and to identify performance trends. Most SLGs are based on response time with a service percentage, such as < 2 seconds 90% of the time. Sometimes SLGs are based on throughput (completions), such as > 100 requests per hour. DBAs investigate missed SLGs so they can resolve the issues and meet SLG goals in the future. The workload event **SLG Response Time** detects missed SLGs.

It is worthwhile to distinguish whether an issue exists in only one workload, a set of common workloads, or all workloads. Individual SLG misses or a set of common workload misses point to a problem that a DBA can solve by using TASM to adjust resource consumption. When most workloads miss their SLGs, this suggests a capacity problem that TASM controls cannot solve.

SLG Throughput

The workload event type **SLG Throughput** detects when a workload misses SLG throughput goals. There are typically two causes of missed throughput goals:

- System overload: If **Arrivals** > **SLG Throughput**, the cause is system overload. The system cannot keep up with requests arriving in the workload, and competing workloads may slow down throughput even more. Only system overload triggers the **SLG Throughput** event in TASM.
- Under-demand: If **Arrivals** < = **SLG Throughput**, the workload missed the SLG because it is not busy enough. An almost idle system can still miss **SLG Throughput**, so qualify the **SLG Throughput** event with **Arrivals** > **SLG Throughput**.

Event Detection Activation and Timing Considerations

Event Evaluation

TASM evaluates events at the **Event Interval** you specify on the **Other** tab of the **General** view. If TASM detects an event at startup, it evaluates the event again at the end of each event interval, except for the component down events, which are only evaluated at TASM activation time.

For example, if the event interval is 1 minute, the events are evaluated at 1:01:00, 1:02:00, and 1:03:00, based on a database-internal timer set when the ruleset was activated. User-defined events can be activated at any time but are not evaluated until the next event interval. For example, an API call activates a user-defined event at 1:01:32 but TASM does not act on the event until the next interval at 1:02:00.

Event Qualification

TASM evaluates event criteria using data accumulated between event intervals. To avoid false event detections, some events must be qualified through a sustained metric reading. There are three methods for qualifying events.

Simple Qualification

Simple qualification requires the event to last for a specified time. The event must continue to exceed the specified threshold until the qualification period is over.

Teradata recommends that you use simple qualification for these event types:

- AWT activity level event types: System events **Available AWTs** and **Flow Control** and Workload event **AWT Wait Time**
- Workload event **Active Requests** (concurrency)

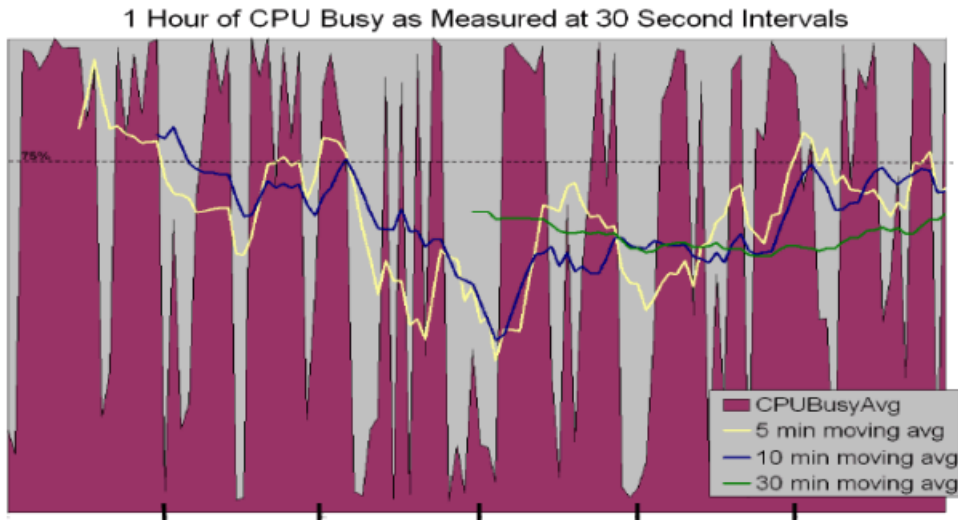
Although TASM offers simple qualification for other event types, Teradata recommends that you use the **Averaging** qualification method for any event not in the preceding list.

Averaging Qualification

The metrics associated with some events fluctuate wildly. To detect this type of event, use the **Averaging** qualification method. This method requires that the moving average of the metric exceed the specified threshold.

To understand the need for **Averaging**, consider the following scenario. ResUsage data is collected at 10-minute intervals. However, event qualifying criteria are collected at every event interval, which can be 5, 10, 30, or 60 seconds. The following figure shows a graph of CPU use measured on an actual production system with 30-second collection intervals. As you can see, CPU use varies greatly. Compare that to the moving averages in the trend lines at 5 minutes, 10 minutes, and 30 minutes.

The smoothing effect of moving averages helps a DBA get a clearer picture of actual CPU use. If a DBA wants to detect if CPU use falls under 75% for at least 10 minutes, **Simple** qualification does not activate the event. CPU use fluctuates too much for that. However, **Averaging** qualification does activate the event.



Use **Averaging** to qualify these system event types:

- CPU Utilization
- CPU Skew
- I/O Usage

Use **Averaging** to qualify these workload event types:

- CPU Utilization
- Arrivals
- SLG Response Time (for workloads created with SLGs)
- SLG Throughput (for workloads created with SLGs)

Immediate Qualification

With **Immediate** qualification, once the metric exceeds the specified threshold, TASM activates the event when it is detected on the next event interval check. This is the only qualification available for these event types:

- Component down events: **Node Down**, **AMP Fatal**, **PE Fatal**, and **Gateway Fatal**
- User-defined events
- Delay queue events (depth and time)

Teradata recommends that you do not use **Immediate** qualification for any other event types.

Resolving Simultaneous Event Detections

You can create multiple events of the same type but with different threshold values. For example, you could create an **Available AWTs** event with a threshold of less than 20 and a second **Available AWTs** event with a threshold of less than 10. When TASM evaluates the events at the end of an event interval, and the available AWTs are 5, both defined events are activated. TASM performs all event actions. For an example of a state matrix with different planned environments and health conditions, see the figure in [State Matrix Basics](#).

If events activate multiple planned environments or health conditions, TASM determines the active state by planned environment precedence and health condition severity. For example, the state matrix has 3 health conditions, in order of highest to lowest severity: **Red**, **Yellow**, and **Normal**. If **< 20 Available AWTs** triggers a change to health condition **Yellow** and **< 10 Available AWTs** triggers a change to health condition **Red**, the **Red** health condition takes precedence and the state changes to the one associated with **Red**.

Event Actions

When you define an event, you also define what TASM should do if the event occurs. Note that TASM logs all detected events to DBC.TDWMEventHistory. TASM logs all detected events and performs notifications, regardless of whether an event causes a health condition or a change in the planned environment.

On the state matrix, **Planned Environments** describe the precedence of planned events. Period events, user-defined events, and event combinations (combinations of the previous two) activate a planned environment.

Health Conditions describe the severity of the system condition. System events, user-defined events (on the health condition axis), workload events, and event combinations (combinations of the previous types of events on the health condition axis) activate a health condition.

TASM changes to the state associated with the highest-precedent active **Health Condition** or **Planned Environment**.

Note:

The **Always** planned environment and the **Normal** health condition cannot have custom events triggers. These are the defaults with the lowest precedence and severity.

To define an event action, edit the state matrix directly or use **Setup Wizard**. On the **Define and Map Unplanned Events** screen, when you add to **Available Events**, you can choose from these actions to occur when the event starts and ends:

- **Post to QTable**
- **Send Alert**
- **Run Program**

Using Alerts or Run Program with Events

Before you can choose the **Send Alert** or **Run Program** actions, you must define those actions in the **Alert Setup** portlet. For more information, see [Using Alerts or Run Program with Exceptions](#).

Use the **Alert Viewer** portlet to see the alerts. Alerts include useful details about what triggered the event.

Using Post to QTable with Events

The **Post to QTable** event action posts a notification to the system notification queue (DBC.SystemQTbl). For more information, see [Using Post to QTable with Exceptions](#).

Key Recommendations for States and Events

Teradata recommends the following best practices for states and events:

- Start with a **Base** state for normal operating conditions and a degraded health condition state for controlling excessive concurrency that has lasted for at least 10 minutes
- After creating a new state, you should review the state-specific values of your filters and throttles.
- Reuse states if possible. Defining too many states may become a maintenance burden.
- Keep in mind that multiple planned environments may be active at the same time, for example, Weekend and Daytime. TASM evaluates planned environments from right to left, so if you want a planned environment to take precedence when more than one could apply, drag that planned environment to the right of the other planned environments in the **States** area of the Viewpoint **Workload Designer** toolbar.

General Parameters

General View

The **General** view in the **Workload Designer** portlet controls everything else related to workload management not covered in the other views. Use tabs in this view to specify the following:

Bypass

Which users, accounts, and profiles to exclude from system-level filters and throttles.

Limits/Reserves

Limit or reserve the resources available to the system, configuring by planned environment.

Other

Specifications for collection and logging intervals, blocker information, timeshare decay, and ordering of the throttle delay queue, among others.

Bypass

You can designate users, accounts and profiles to be exempt from TASM system-level filtering and throttling. For example, you might give a DBA bypass privileges so that the DBA can always access the system for troubleshooting. Users TDWM and DBC are, by default, bypassed users, so they are not affected by system-level throttles.

Note:

TASM still manages bypassed requests as part of a workload, including workload throttling.

Tip:

If a user is marked as bypassed, the user will not be filtered or throttled at the system level. Use this setting very selectively.

Limits and Reserves

You can define system-wide resource limits and reserves in Viewpoint **Workload Designer** in the **Limits/Reserves** tab of the **General** area. These values can be different for each planned environment.

Setting	Description
CPU Limit	By default, a system uses 100% of CPU resources. However, right after a system upgrade, you could use a lower percentage to mimic response times on the old system.

Setting	Description
	Using a lower CPU limit keeps user expectations in check until the new system has a normal load. You can gradually increase the CPU limit as data and workloads are added to the new system.
I/O Limit	By default, a system uses 100% of I/O resources. Teradata recommends that the I/O Limit and the CPU Limit always be set to the same value, for example, if CPU Limit is 85%, set I/O Limit to 85%.
Reserved AWTs under Tactical Reserves	<p>This setting reserves AMP Worker Tasks for use by Tactical workloads and SLG Tier 1 expedited workloads. Teradata recommends using this setting only if you are sure that your system has an ongoing AWT shortage. To determine this, take one of the following actions:</p> <ul style="list-style-type: none"> • Enable logging for the ResUsageSAWT table, which shows patterns in AWT use. • Monitor current AWT in-use counts with the either of these Viewpoint portlets: Metrics Analysis or Metric Heatmap. <p>If you find an ongoing AWT shortage, try throttling lower-priority workloads before using the Reserved AWTs option. This option removes more AWTs from general use than you specify. Because there are multiple reserved pools set up based on this reserve count, the number of AWTs removed from the general pool equals (reserve count *2) +2. If you select a reserve of 5 AWTs, 12 AWTs are removed from the general pool: (5*2) +2 =12.</p>
Max AWTs for new tactical work under Tactical Reserves	This setting limits the number of AWTs that can process new Tactical work. Teradata recommends using the default of 50 if you specify Reserved AWTs .

Other Settings

Intervals

You can set the following intervals on the **Other** tab of the **General** view:

Event Interval

How often TASM evaluates all of the events defined in the state matrix. The default is 60 seconds.

Flex Throttle Action Interval

How often TASM evaluates system resource availability. The default is 60 seconds.

Dashboard Interval

How often TASM collects data about workloads, including the following metrics:

- Arrivals
- Completions
- Delays

- Exceptions
- Average response time
- CPU use
- I/O use
- The number of requests that meet their SLGs

This data is displayed real-time in the Viewpoint **Workload Monitor** portlet. It is also logged in the long-term repository, TDWMSummaryLog. Teradata recommends setting this interval to 60 seconds to match the default refresh rate of the Viewpoint **Workload Monitor** portlet.

Logging Interval

How often TASM flushes log files from memory to disk. The following log tables are refreshed at this interval:

- TDWMSummaryLog
- TDWMEExceptionLog
- TDWMEEventLog
- TDWMEEventHistoryLog

If log memory cache fills up before the logging interval ends, Vantage flushes the log file to disk immediately. The **Logging Interval** specifies the maximum time between memory flushes.

Exception Interval

How often TASM checks for exceptions. The default is 60 seconds, which is a reasonable interval for detecting a long-running step.

Both the **Dashboard Interval** and the **Logging Interval** must be multiples of the **Event Interval**. Also, the **Logging Interval** must be a multiple of the **Dashboard Interval**. This ensures a smooth roll-up of summary data to the dashboard data and the logging data.

Note:

The **Exception Interval** affects only exception checks done during a request, when the request duration exceeds the exception checking interval. The **Exception Interval** does not affect the exception checks TASM does at the end of each request step.

Tip:

Teradata recommends that the **Exception Interval** be less than or equal to the session rate, which is the rate for updating session-level statistics in memory. The session rate is controlled by the PM/API request SET SESSION RATE.

If using skew difference, beware of an issue in detection accuracy: Whenever multiple applications (besides TASM asynchronous exception checking) issue MONITOR SESSION commands, the internal

collection cache is flushed and accumulations restart at the shortest interval being used. If Teradata Viewpoint is set to refresh (submit MONITOR SESSION) every 30 seconds and the exception interval is set to 60 seconds, Viewpoint will receive new data and reset the cache every 30 seconds. When TASM issues the MONITOR SESSION command, it will contain not 60 seconds, but just 30 seconds of accumulated data.

Tip:

In order to keep the MONITOR SESSION data TASM uses as complete as possible, Teradata recommends that other PMPC applications that use MONITOR SESSION do so at an interval greater than the exception interval. That way, the other PMPC applications will always get the PMPC statistics TASM collects, maintaining the accuracy of TASM in computing skew difference values.

Note:

An additional timer, DBQLFlushRate, defined in the DBS Control utility General fields, tells the system how often to flush *all* DBQL tables (including step, object, and SQL tables). The **Logging Interval** causes DBQLogTbl to be flushed at an additional rate.

Blocker

If a single-statement transaction is delayed by a throttle, it does not receive resources, such as database locks and AMP worker tasks, until it is released from the queue and starts to run. However, a request that is part of a transaction (multiple requests between BT/ET) may hold onto resources while waiting in the delay queue. TASM can automatically detect the following sequence of events:

1. A transaction starts to run.
2. One of the first requests in that transaction obtains locks, which will not be released until the entire transaction completes.
3. The system is busy, so a throttle delays a later request in the transaction.
4. Requests that are running now need the locks held by the request in the delay queue. These active requests cannot complete because they cannot get the locks they need.
5. TASM cannot release the request in the delay queue because the active requests are not completing.

Sometimes blocked requests resolve themselves. TASM can also act when a throttle delays one request in a transaction. The **Block Cycles** parameter together with the **Exception Interval** parameter specifies how long TASM should wait before acting on a blocked request.

The **Block Action** parameter controls what TASM does when the number of **Block Cycles** has been met. TASM can act on a request in the delay queue in one of these ways:

- **Log** that it is holding a resource needed by a running request
- **Abort** it because it is holding a resource needed by a running request

- **Release** it to run

If you select **Abort** or **Release**, TASM logs that action to DBC.TDWMEventLog.

By default, **Block Cycles** is set to 1 and **Block Action** is set to **Release**. If a blocked request occurs, the number of **Block Cycles** should be enough to let the system try to resolve the block normally. After the waiting interval, TASM lets the blocking request run so it can free locks for other requests. If **Exception Interval** is set to 60 seconds and **Block Cycles** is set to 2, two 60-second cycles (2 minutes) must pass before TASM acts. If **Block Cycles** is set to **Off**, TASM does not resolve blocked requests.

Note:

One drawback to TASM blocked request resolution using the **Release** option is that it makes concurrency limits less strict. For example, if the throttle limit is 5, 6 or more requests may be running. When a throttle count is over limit, new requests stay in the delay queue longer waiting for the throttle count to decrease. If many transactions use TASM blocked request resolution, some work could be so significantly delayed in starting that overall workload management is affected. Evaluate TASM blocked request resolution carefully and monitor the impact of it on other work.

Activation

The **Activation** Parameter controls which features are enabled when you activate the ruleset. Select one or both of these options:

- **Filters and Utility Sessions**
- **System Throttles and Session Control**

Utility Sessions controls the number of sessions used by a specific utility. Specify values for **Utility Sessions** in Viewpoint **Workload Designer** in the **Sessions** area under the **Utility Sessions** tab.

Session Control throttles the number of sessions logons by classification criteria. Specify values for **Session Control** in Viewpoint **Workload Designer** in the **Sessions** area under the **Query Sessions** tab.

Timeshare Decay

Many customers use estimated processing time to prioritize shorter requests. To do this, customers usually define three workloads for each account, based on CPU usage:

- Short
- Medium
- Long

Customers also create exceptions on the Short and Medium workloads so that longer-running requests move to a lower-priority workload.

Another way to prioritize shorter requests is to enable the **Timeshare Decay** option. This option decreases the resources available to longer-running requests. This option applies to all requests running in the Timeshare tier; it does not apply to workloads in the Tactical or SLG tiers. TIWM does not include exceptions, but DBAs can use the **Timeshare Decay** option to provide similar functions in demoting requests.

The **Timeshare Decay** works as follows:

1. A request starts with the resource access rate assigned to it based on workload and workload management method (tier).
2. TASM halves the resources available to the request on that node after the request consumes 10 seconds of CPU or 100 MB of I/O on that node.
3. TASM again halves the resources available to the request on that node after the request consumes 200 seconds of CPU or 10,000 MB of I/O on that node. This access rate remains constant until the request ends.

Note:

TASM performs timeshare decay on each node independently. This means that a request can be running at different decay levels on different nodes.

Before deciding to use **Timeshare Decay**, consider these tradeoffs:

- All Timeshare tier workloads are affected. TASM cannot target decay to specific workloads.
 - The thresholds that trigger decay are the same for all workloads in all access levels. Top is treated the same as Low.
 - If many Low requests experience decay, they may receive so few resources that they hold locks and AMP worker tasks for unreasonably long times.
 - If most of the requests in Timeshare experience automatic decay, most of the requests have the same relative priority to each other as they did before the decay was applied.
-

Tip:

For systems with TASM, workload exceptions provide more robust and tunable options to achieve the same goal as **Timeshare Decay** by using the change workload action. For TASM systems, workload exceptions are more predictable and measurable. **Timeshare Decay** is appropriate for TIWM systems, which do not have workload exceptions.

If you enable query logging, the level of decay for a request is logged in the CPUDecayLevel and IODecayLevel fields of DBQLogTbl.

Prevent Mid-transaction Throttle Delays

The **Do not delay queries within transactions that are holding locks higher than 'access'** option on the **Other** tab of the **General** view overrides throttle limits when a request is part of a transaction that

holds a lock higher than access level. Like the **Blocker** options, this option prevents a request in the delay queue from holding locks needed by active requests.

This option applies to all request throttles. Because it exempts certain requests from throttles, this option can cause the number of concurrent requests to exceed the throttle limit. Evaluate how use of this option affects your site. The TDWM Statistics API returns the number of active requests that are running solely due to this option. In addition, the following DBQLogTbl columns can help you to assess the impact of this feature on throttle limits.

DBQLogTbl Field	Description
LockLevel	The highest-level lock obtained when the request ran. This lock is from a previous request in the transaction.
ThrottleBypassed	Indicates if this request ran solely due to the Prevent Mid-transaction Throttle Delays option. This flag is not set if running the request did not override the throttle limit.
TnxUniq	Used in conjunction with the userid and procid to uniquely identify the transaction under which this request ran.

Order the Throttle Delay Queue

In the **Other** tab of the **General** view, there are two options for ordering the throttle delay queue:

By time delayed (the default)

This option orders the delayed request by the time delayed, regardless of the owning workload.

By workload priority

In this option, TASM calculates the workload priority based on the workload management method (tier) assigned to the workload. Requests in the delay queue are ordered from high to low based on workload priority. Ties are ordered by start time.

For more information on workload management methods and the priorities assigned to them, see [Workload Priority Management](#) and [Workload Priority Order](#).

Utility Limits

For a discussion of the **Support increased MLOADX job limits and increased AWT resource limits** option on the **Other** tab of the **General** view, see [Default AWT Resource Limits](#).

Define Available AWTs

TDWM monitors system resources and can trigger events and actions if resources reach a critical level. One system event, **Available AWTs**, detects a shortage of AMP Worker Tasks. An AWT shortage may indicate an overly busy system.

In the **Other** tab of the **General** view, you can define the type of AWT availability you want TDWM to monitor. Choose one of these options to define **Available AWTs**:

Option	Description
AWTs available for the WorkNew (Work00) work type	The default. AWTs reserved for beginning new work.
AWTs available in the unreserved pool for use by any work type	Unreserved AWTs available for any type of work, including new work.

Key Recommendations for General Parameters

For an initial implementation, Teradata recommends that you use the defaults on the **Bypass**, **Limits/Reserves**, and **Other** tabs. Teradata also has the following recommendations for the **General Parameters** area.

Setting	Recommendation
Blocker	<ul style="list-style-type: none"> For Block Action, try Release. It allows for some concurrency management while causing minimal disruption to running requests. You can try another setting later if needed. For Block Cycles, use 1. Create views with the LOCKING FOR ACCESS modifier. This allows read-only users and users who modify the data to have concurrent access. Use of access locks prevents deadlocks and improves performance.
Bypass	A typical practice is to set Bypass for BAR work and critical DBA activities. Bypassing BAR work does not bypass utility session limits.
Intervals	<ul style="list-style-type: none"> Teradata recommends that you use the default interval settings unless you have a reason not to. Set the Event Interval to no lower than 30 seconds for systems with more than 500 AMPs. Set the Exception Interval to equal to or less than the MONITOR SESSION rate. Do not set the Exception Interval to lower than 6 seconds.

Note:

Users DBC and TDWM are included in the bypass list by default. System filters and throttles do not apply to them. If other users are affected by system filters and throttles in a way that does not meet workload management goals, adjust system filter and throttle classification criteria so that these users, account names/strings, or profiles are not affected. Avoid using the bypass list to make exceptions for users because that may result in situations where the necessary system filters and throttles are not applied.

Advanced SQL Engine Analytic Functions and Workload Management

TASM with Advanced SQL Engine Analytic Functions

Advanced SQL Engine analytic functions can be memory- and compute-intensive, depending on function parameters and input table sizes. You can limit impact on other workloads by using TASM throttles to limit concurrency and memory.

Table Operators for Workload Management with Advanced SQL Engine Functions

This table describes situations where each function can impact workload and provides the related table operator names to use with TASM.

Serial Number	Function	When Function Impacts Workload	Table Operator Name for TASM
1	NaiveBayesPredict	As model size increases.	NAIVEBAYESPREDICT
2	DecisionTreePredict	As model size increases.	DECISIONTREEPREDICT
3	DecisionForestPredict	As model size increases.	DECISIONFORESTPREDICT
4	GLMPredict	As model size increases.	GLMPREDICT
5	SVMSParsePredict	As model size increases.	SVMSPARSEPREDICT
6	NaiveBayesTextClassifierPredict	As model size increases. As input table partition size increases. When ModelType is Bernoulli.	NAIVEBAYESTEXTCLASSIFIERPREDI* (Note * at end.)
7	MovingAverage	When MAVgType is simple, triangular, or weighted: As window size and number of target columns increase.	MOVINGAVERAGE

Serial Number	Function	When Function Impacts Workload	Table Operator Name for TASM
8	NGramSplitter	As gram size and document size increase.	NGRAMSPLITTER
9	Pack	As number of input columns increases.	PACK
10	StringSimilarity	As string size increases.	STRINGSIMILARITY
11	Unpack	Not applicable	UNPACK
12	Antiselect	Not applicable	ANTISELECT
13	nPath	<p>When input data partition is huge and number of ON clauses increases.</p> <p>When Mode is NONOVERLAPPING and Pattern includes end anchor operator (\$) but not start anchor operator (^).</p> <p>When Mode is OVERLAPPING, and Pattern does not include start anchor operator.</p> <p>When first symbol in Pattern can match infinite number of input rows.</p> <p>When conditions are imposed using Filter.</p>	NPATH
14	Attribution	As partition size or WindowSize increases.	ATTRIBUTION
15	Sessionize	Not applicable	SESSIONIZE

Creating a Viewpoint User for Data Collection

1. Run the following script on the Advanced SQL Engine:

```
.logon dbc,dbc

create user viewpoint as perm=2e9
password=viewpoint;
```

```
create user console as perm=50000, spool=50000, account='$H-remoteconsole-
use', password=console, fallback;
.logoff
```

For more information about creating Viewpoint users, see *Teradata® Viewpoint Installation, Configuration, and Upgrade Guide for Customers*, B035-2207.

Adding an Advanced SQL Engine to the Monitored Systems Portlet

For details about using Teradata Viewpoint portlets, see *Teradata® Viewpoint User Guide*, B035-2206.

Monitored Systems is an administration portlet.

1. In the Viewpoint **Monitored Systems** portlet, add an Advanced SQL Engine using the following values.

Field	Values
Enable system	Check the box.
Login credentials	<ul style="list-style-type: none"> • User name: tdwm Password: tdwadmin • User name: viewpoint Passowrd: viewpoint
Grant	For each user name, grant privileges for Viewpoint data collection.
Collectors	Select Enable Data Collectors(Configure in Data Collectors) .
Enhanced TASM Functions	Select Enable this option if your Teradata system has license entitlement to TASM .

Example: Creating a TASM Ruleset to Limit SQLE Analytic Function Concurrency

For details about using Teradata Viewpoint portlets, see *Teradata® Viewpoint User Guide*, B035-2206.

Workload Designer is a workload management portlet.

1. In the Viewpoint **Workload Designer** portlet, create a ruleset using the following values.

Field	Value
Name	SQLAnalyticFunctionThrottle
Description	Ruleset - throttle to limit SQLE Analytic functions concurrency to 4

2. Create a new system throttle using the following values.

Field	Value
Name	CPU and Memory Intensive.
Description	Set concurrency limit 4 for all SQLE Analytic Functions.
Rule Type	Select Collective: one queue for all matched queries.

3. Set classification criteria for a system throttle using the following values.

Field	Values
Add Classification Criteria	Select Target .
Target Type	Select Function .
Database	Select TD_SYSFNLIB .
Functions	Select and include using the Include button: DECISIONFORESTPREDICT DECISIONTREEPREDICT GLMPREDICT MOVINGAVERAGE NAIVEBAYESPREDICT NGRAMSPLITTER SVMSPARSEPREDICT ATTRIBUTION NPATH PACK SESSIONIZE STRINGSIMILARITY UNPACK ANTISELECT
Included Functions	Due to character length, the NAIVEBAYESTEXTCLASSIFIERPREDICT function is abbreviated using a wildcard: NAIVEBAYESTEXTCLASSIFIERPREDI* and cannot be included from the Functions list. Instead, include it directly.

4. Add state-specific settings using the following values.
 For this example, we use the value 4.

Field	Values
Concurrency Limit	<ul style="list-style-type: none"> Select the blank field and type 4. Select Delay.

5. Activate the ruleset.

Checking Query Status

- Use the **Query Monitor** portlet.
See *Teradata® Viewpoint User Guide*, B035-2206.
- Check the status of queries on a given node using the `tdwmdmp` command at the command prompt.
Sample output:

Delayed Requests:

** WLC of 0 means the actual workload is not throttled **

HOST NAME /QUERYID	SESSION	INT REQ DELAYD	TYP SWGU	DEP OV BL	USER NAME PRIO	WLC FLEXQLFY	RULE TYPE (ID)
1 Names 307191689474943282	488612	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307181689474968210	488611	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307181689474968214	488613	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307181689474968216	488615	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307191689474943283	488610	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307191689474943284	488608	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307181689474968215	488609	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307191689474943285	488614	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1 Names 307191689474943287	488607	4 13	QRY 1000		1 NBTCP 1 0 0	0	QRY (80)Tblop
1	488616	4	QRY		1 NBTCP	0	QRY (80)Tblop

```
Names          13          1000   1  0    0
307191689474943286
```

Active Requests:

```
HOST  SESSION      INT REQ  TYP  DEP  USER NAME      WLC  RULE TYPE (ID)
NAME          DELAYD  SWGU  OV BL  PRI0   FLEXQLFY
/QUERYID
```

Rules matching Delayed Requests but below threshold (Non-impacting):



Managing Workloads

Monitoring Workload Management Activity

Using Viewpoint Workload Monitor

The Viewpoint **Workload Monitor** portlet tracks workload management activity in Vantage. It gives users multiple views of workloads, including the following information:

- Allocation of system resources per workload
- Details about requests processed, delayed, and rejected
- Evaluation of skew
- Details about each active request
- Service level goals for workloads and how actual performance compares to those goals

For more information, see *Teradata® Viewpoint User Guide*, B035-2206.

Using Viewpoint Workload Health

The **Workload Health** portlet displays workload management activity in Vantage for one system at a time. You choose which system, workloads, and metrics to monitor. **Filter** and **Sort** options allow you to customize the information displayed. Data in the **Workload Health** portlet is refreshed every minute and displays workloads that have the following characteristics:

- Completed processing according to their response-time service-level goals
- Missed their response-time service-level goals
- Are inactive
- Are disabled
- Have no defined response-time service-level goals

The **Health Details** view displays detailed metrics for an individual workload.

Using Log Files

About the Log Files

The following Data Dictionary tables and views in database DBC contain TASM information. Teradata recommends using the views to access data in the tables to make sure that the tables are not accidentally modified or deleted. For information about the views, see *Teradata Vantage™ - Data Dictionary*, B035-1092.

DBQLogTbl

The main database query logging table, DBQLogTbl, contains data about all logged requests. The Teradata Dynamic Workload Management view of this table is QryLogTDWMV. Teradata recommends that you begin query logging before you implement TASM so that you understand the requests coming into the system. For information about database query logging (DBQL), see *Teradata Vantage™ - Database Administration*, B035-1093.

TDWMEventLog

The TDWMEventLog table and the view that corresponds to it, QryLogEventsV, contain information on the following events:

- Ruleset activations, including times when activations and optimizations are made
- State changes, including times and details
- When a request is moved between workloads due to an exception or manual (DBA) intervention

TDWMSummaryLog

The TDWMSummaryLog table and the view that corresponds to it, QryLogTDWMSumV, contain data about requests in a workload, summarized over a collection period. Users can define the logging interval in Viewpoint **Workload Designer**, in the **General** area. See the **Intervals** section on the **Other** tab.

TDWMEventHistory

The TDWMEventHistory table and the view that corresponds to it, QryLogEventHisV, contain information about workload and system events and the values that lead up to a state change.

The table is structured so that the relationship between events can be extracted as a set. Several independent events may help determine the current state. To make this relationship clear, the table lists related events in chronological order under the same timestamp.

For example, when an event occurs, TASM logs it immediately. If this event is part of an event combination that causes a planned environment change, TASM logs the planned environment change and the details of all the events in the event combination. If the planned environment change then triggers a state change, TASM logs the state change under the same timestamp. The seqno (sequence number) field shows the order in which the events occurred. The following is a sample SQL request that extracts events in chronological order.

```
SELECT entryts,
SUBSTR(entrykind,1,10) "kind",
SUBSTR (entryname,1,20) "name",
CAST (eventvalue as float format '999.9999') "evt value",
CAST (lastvalue as float format '999.9999') "last value",
spare2 "spare Int",
SUBSTR (activity,1,10) "activity id",
SUBSTR (activityname,1,20) "act name", seqno
FROM tdwmeventhistory order by entryts, seqno;
```

Here is an example of another approach to accessing the TDWMEventHistory log, using recursion to show what caused the system to enter the RED state:

```
WITH RECURSIVE
CausalAnalysis(EntryTS,
EntryKind, EntryID, EntryName, Activity,Activityid) AS
(
  SELECT EntryTS, EntryKind, EntryID, EntryName, Activity, Activityid
  FROM DBC.TDWMEventHistory
  WHERE EntryKind = 'SYSCON' AND EntryName = 'RED' AND Activity = 'ACTIVE'
  UNION ALL
  SELECT Cause.EntryTS,Cause.EntryKind,Cause.EntryID,
    Cause.EntryName,Cause.Activity,Cause.Activityid
  FROM CausalAnalysis Condition INNER JOIN DBC.TDWMEventHistory Cause
  ON Condition.EntryKind = Cause.Activity AND
    Condition.EntryID = Cause.Activityid)
SELECT * FROM CausalAnalysis
ORDER BY 1 DESC;
```

EntryTS	EntryKind	EntryID	EntryName	Activity	ActivityId
2006-03-22 14:20:49.12	SYSCON	30	RED	ACTIVE	0
2006-03-22 14:19:49.06	EXPRESS	20	AWT&NODE	SYSCON	30
2006-03-22 14:17:48.53	EVENT	11	NODE DOWN	EXPRESS	20
2006-03-22 14:16:48.51	EVENT	10	LOW AWTs	EXPRESS	20

TDWMEExceptionLog

The TDWMEExceptionLog table and the view that corresponds to it, QryLogExceptionsV, contain a row for each exception that TASM detects for a request, including filter, throttle, and workload exceptions. There could be several rows for one request. The information includes the action that TASM performs when it detects an exception. TASM logs the SQL in the request that caused an exception in DBQLSQLTbl. The corresponding view is QryLogSQLV.

Workload Activity Logging and DBQL Logging Comparison

TASM workload activity logging is automatic and a useful complement to DBQL request logging. At each logging interval, TASM writes summarized details about each workload to the TDWMSummaryLog table. TASM collects the same summary data in memory and uses it in the **Workload Monitor** portlet. This data includes the following types:

- Arrival rate
- Response time
- CPU time
- Request counts for these categories: Active, completed, failed due to error, rejected, delayed (and on which level of throttles), encountered exceptions, requests that move into or out of the workload, and met SLGs.

DBAs can either use the Viewpoint **Workload Monitor** portlet or develop their own custom reports to see the historical data about workloads that is in the TDWMSummaryLog or various DBQL tables.

No matter what DBQL logging options you use, if TASM finds an exception, it logs information about it in both the exception log and DBQLLogTbl. This way, a DBA can follow up on the information.

TASM and TIWM Features

A Comparison of TASM and TIWM Features

The following table explains the differences between TASM and TIWM resource and prioritization management in this release.

Item	TIWM SLES 11	TASM SLES 11
Resource management	By CPU and I/O	By CPU and I/O
Resource limits (hard limits)	By CPU and I/O	By CPU and I/O
Priority groups	Tactical and Timeshare tiers	Tactical, SLG tiers, and Timeshare
Priority weighting	Automatic	Configurable

The following table explains the differences between TASM and TIWM management features in Vantage in this release.

Item	TIWM SLES 11	TASM SLES 11
Viewpoint user interface	Yes	Yes
System filters	Yes	Yes
System throttles	Yes	Yes
Workload classification	Full	Full
Workload throttles	Yes	Yes
Workload service-level goal reports	Yes	Yes
Workload exception processing	Tactical: partial Timeshare: optional decay	Full
Workload exception actions	Full/less access to resources	Full
Event management by date, time, resources, and conditions	By date/time only	Yes
Virtual partitions	No	Yes
SLG tiers	No	Yes
Reserved AMP worker tasks and expedited Tactical workloads	Tactical workloads	Tactical workloads and SLG Tier 1 workloads

Additional Information

Teradata Links

Link	Description
https://docs.teradata.com/	Search Teradata Documentation, customize content to your needs, and download PDFs. Customers: Log in to access Orange Books.
https://support.teradata.com	One-stop source for Teradata community support, software downloads, and product information. Log in for customer access to: <ul style="list-style-type: none">• Community support• Software updates• Knowledge articles
https://www.teradata.com/University/Overview	Teradata education network
https://support.teradata.com/community	Link to Teradata community