KPI Metrics Overview

An Open Source Asset for use with TIBCO® Data Virtualization

Project Name	AS Assets KPI Metrics
Document Location	This document is only valid on the day it was printed. The source of the document will be found in the ASAssets_KPI folder (https://github.com/TIBCOSoftware)
Purpose	Self-paced instructional

www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304

Tel: +1 650-846-1000
+1 800-420-8450
Fax: +1 650-846-1005

Table of Contents

1 Introduction

Purpose

The purpose of this document is to provide an overview for the AS Assets KPI Metrics.

Audience

This document is intended to provide guidance for the following users:

- Data Virtualization Administrators – provides a guide for installation.
- Architects – provides the KPImetrics architecture.
- Data professionals – provides background on the published views and usage.
- Operations users – provides insight into triggers and procedures that are executed.
- Project Managers – provides general information on KPImetrics.

References

Product references are shown below. Any references to CIS or DV refer to the current TIBCO® Data Virtualization.

- TIBCO® Data Virtualization was formerly known as
    - Cisco Data Virtualization (DV)
    - Composite Information Server (CIS)

2 Overview

Overview

This document outlines the installation, configuration and use of the KPImetrics data collection asset for the **Data Virtualization (DV)** instance. The KPImetrics is an add-on to the DV metrics built-in capability. The KPImetrics module connects system metrics and usage data to monitoring of resource utilization and system capacity. It provides aggregation of data. It provides a more efficient strategy for processing large amounts of KPI data.

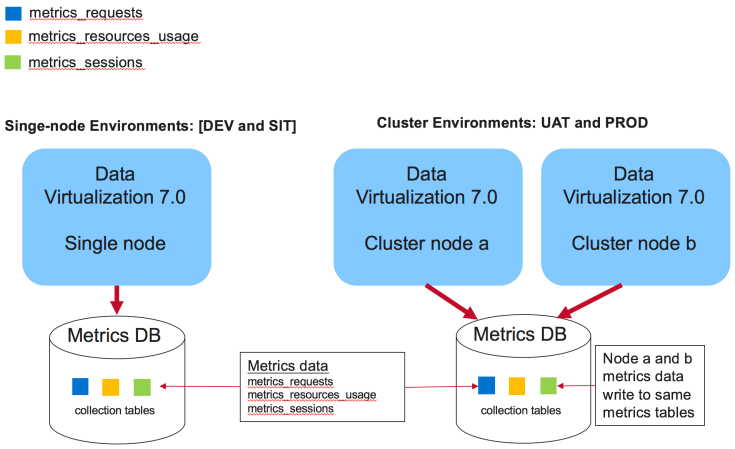
KPImetrics Metadata module provides a scheduled collection of resource metadata including:

- Resources, Columns, Lineage, Datasources, Row-based security, Column-based security, and Privileges.
- Provides a strategy for enforcing required columns and view layer invocations.
- Provides a set of constant tables to define layers, paths and enforcement rules.

KPImetrics Metadata allows the users to report on who modified what resource last and how many resources are in what layers in additiona to a whole host of other metadata related queries.

The **out-of-the-box DV Metrics** captures the following information:

- Session information
  - Client session information but does not contain user unless correlated with resource usage.
  - Client throughput and duration.
- Request information
  - Captures the request from a client such as the request (SQL statement), duration, rows affected, max memory, and max disk usage.
- Resource usage information:
  - Resource accessed with start/end time and user.
  - Data source path and type and resource path and type.
  - The resource kind which identifies whether this was a “system” or “user defined” request.
- Diagram of the out-of-the-box DV Metrics is shown below:



KPImetrics Advantage

The biggest advantage for using **ASAssets KPImetrics** in addition to the DV metrics is as follows:

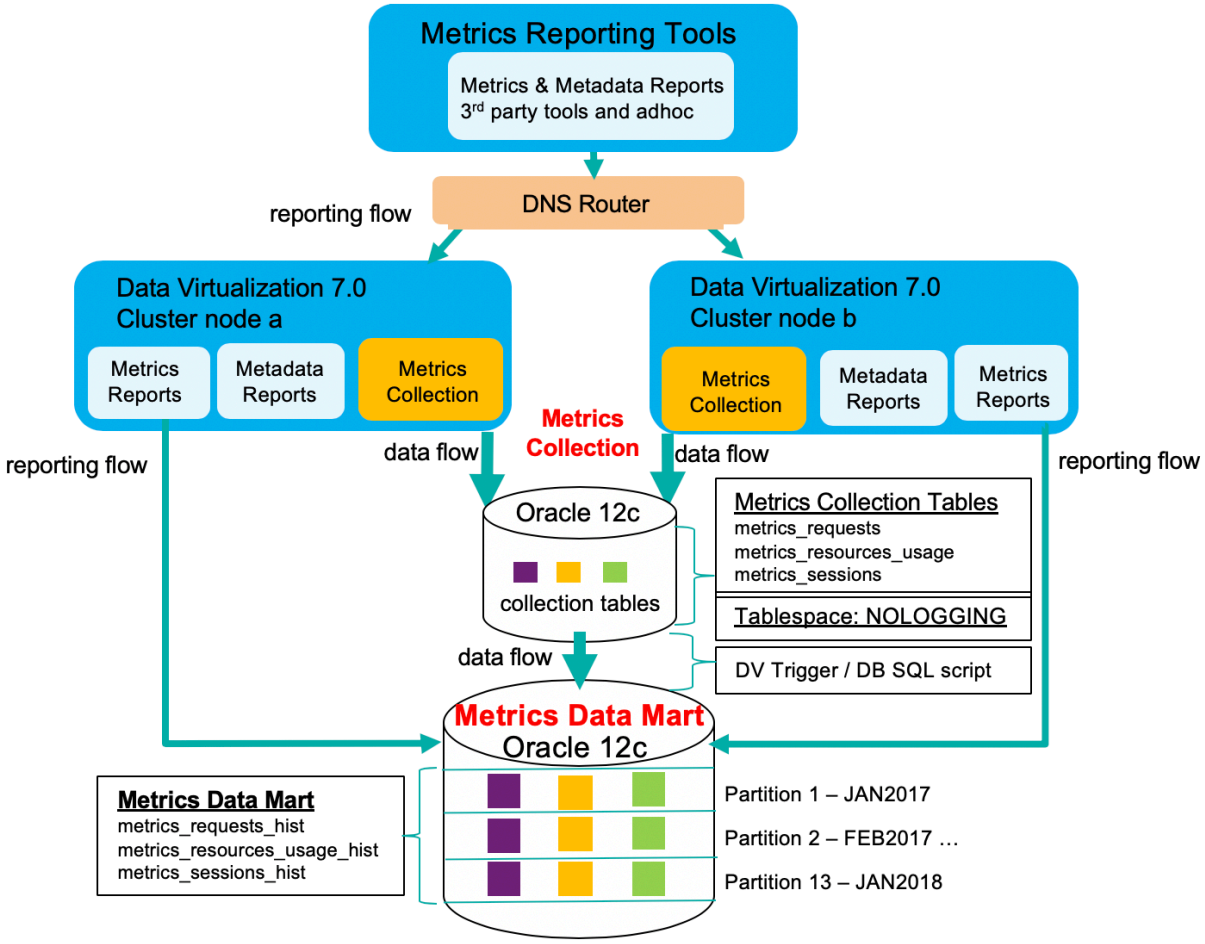
- Reporting strategy
  - Consider the DV metrics out-of-the-box as collection or stage tables. They are not designed for doing reporting unless the content is very, very small. Metrics is a true reporting application that needs to be treated like one from a database architecture point-of-view in order to support high-volume applications. The KPImetrics implementation has collection tables, stage tables and data mart tables.
  - Why Collection/Stage Tables
    - Provides a mechanism for filtering out unwanted rows
    - Faster insert/delete by keeping overall size small
    - Faster insert/delete with no indexes
    - Minimize DV workload for insert/delete
    - Stage tables are used to prune and augment the data prior to inserting into the history tables.
    - Ability to create no logging tables and tablespace to minimize REDO and TEMP space issues when using Oracle
  - Why Data Mart Tables
    - Only transfer collection data that is wanted and filter out unwanted rows to minimize the overall data mart size.

        - Ability to create indexes to improve performance
        - Ability to partition table based on date to achieve the following:
            - Ability to purge data 1 month at a time – quickly with no delete (logging) overhead
            - Ability to query table in parallel to improve performance
    - KPImetrics provides published views that are tailor-made for reporting tools to quickly create different reports based on the type of view being published. In other words, the user of KPImetrics has a faster time-to-market due to the aggregation and views that have been provided out-of-the-box than if they had to build the report views by themselves.

- KPImetrics collects additional data not collected by DV metrics allowing performance, usage and capacity planning reports may be generated on demand.
    - User resource utilization:
        - Organizational hierarchy of LDAP user
        - Number of requests executed by user over time
        - Data throughput for resources and frequency of their usage
    - Server resource utilization
        - Memory and CPU Utilization of physical server
        - Overall throughput of data [Memory, IO, Disk, Caches, Data source]
        - Number of sessions and requests executed on DV server
    - Developer resource utilization
        - Resource ownership for a developer
        - Reusability/Dependency of resources
        - Identification of orphaned resources. (published but not used)

KPImetrics Reporting Strategy

The following diagram depicts a “KPImetrics Reporting Strategy” utilizing the primary database “Oracle 11g/12c” as the KPImetrics repository. This picture depicts what was described in the above section titled “Reporting strategy”. This strategy may be adapted to other supported RDBMS in whole or in part as functionality is mapped.



Reporting Flow

The reporting flow represents the fact that reporting tools will query (read-only) various published KPImetrics views from the historic data mart tables. Reporting is how users view the data that has been collected over time.

Data Flow

The data flow represents data flowing from the DV metrics collection tables to the historic data mart tables. Additionally, this represents KPImetrics triggers/scripts that execute on a periodic basis and process other DV non-metrics type of data such as IO, CPU, memory, data sources and SQL parsing. Data flow is a constant data collection capability driven by the out-of-the-box DV metrics.

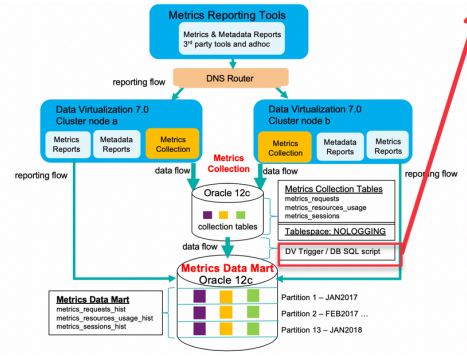
Collection Data Processing

The KPImetrics module integrates with the DV out-of-the-box metrics capability. It uses the same table names as the out-of-the-box solution to represent the collection tables [*metrics_sessions, metrics_requests, metrics_resources_usage*] and stage tables [*metrics_sessions_stg, metrics_requests_stg, metrics_resources_usage_stg*]. However, it is recommended to use the KPImetrics DDL to create the tables especially with Oracle so that they can be created in a "NOLOGGING" tablespace. Unfortunately, SQL Server has limited capabilities for a similar no logging features and is not currently supported. Refer to the database setup section for details.

The following diagram describes how the data flows from the collection tables into the history tables. A scheduled job runs every 2 hours to transfer records and delete from the collection tables to keep them small.

	No Logging	Logging
metrics_sessions	→ XFER Job [metrics_sessions_stg]	→ metrics_sessions_hist
metrics_requests	→ XFER Job [metrics_requests_stg]	→ metrics_requests_hist
metrics_resources_usage	→ XFER Job [metrics_resources_usage_stg]	→ metrics_resources_usage_hist
metrics_resources_usage_hist	→ XFER Job	→

A diagram of the DV Trigger and native database script is provided below.



Supporting Tables
METRICS_JOB_DETAILS – log data transfer progress
METRICS_JOB_FILTERS – lookup filters
METRICS_JOB_ENVIRONMENTS – lookup environments
METRICS_LDAP_PERSON – lookup person info
METRICS_ALL_USERS – lookup DV users
METRICS_ALL_RESOURCES – lookup DV resources

Metrics Collection Tables
metrics_requests
metrics_resources_usage
metrics_sessions

Metrics Data Mart Tables
metrics_requests_hist
metrics_resources_usage_hist
metrics_sessions_hist

DV Trigger / DB SQL script - P_METRICS_ALL_TABLES('N','DEV1','hostname', 9400)
1. Get metrics_sessions: MAX(logintime) and MAX(sessionid)
2. Get metrics_resources_usage: MAX(starttime) and MAX(requestid)
3. Get metrics_requests: MAX(starttime) and MAX(requestid)

PROCESS: metrics_sessions collection table
1. Insert 'INITIALIZE' status into METRICS_JOB_DETAILS
2. Insert/Update metrics_sessions_stg
3. Insert into metrics_sessions_hist select from metrics_sessions_stg
4. Execute table statistics in metrics_sessions_hist
5. [**metrics_session collection table clean-up**] Delete from metrics_sessions
6. Update METRICS_JOB_DETAILS set 'COMPLETE" status

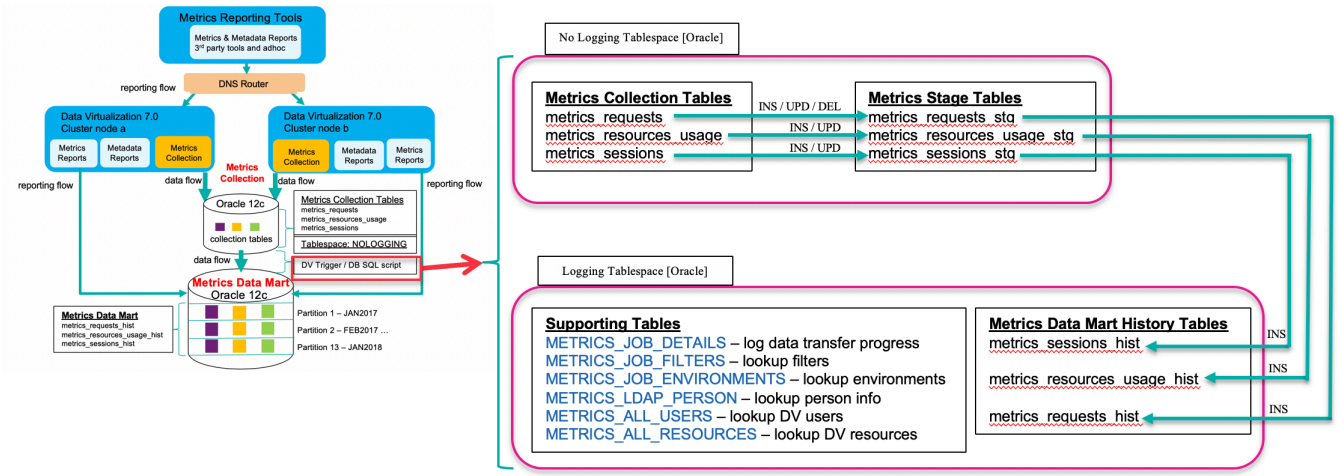
PROCESS: metrics_resources_usage collection
1. Insert 'INITIALIZE' status into METRICS_JOB_DETAILS
2. Insert/Update metrics_resources_usage_stg
3. [**Filter out user-defined job filters**] Insert from metrics_resources_usage where (user, domain, resourcekind) not in METRICS_JOB_FILTERS
4. Insert into metrics_resources_usage_hist select from metrics_resources_usage_stg
5. Execute table statistics in metrics_resources_usage_hist
6. Update METRICS_JOB_DETAILS set 'COMPLETE" status

PROCESS: metrics_requests collection
1. Insert 'INITIALIZE' status into METRICS_JOB_DETAILS
2. Insert/Update metrics_requests_stg
3. [**Filter out non-correlated rows**] Insert from metrics_requests where exist in metrics_resources_usage
4. Insert into metrics_requests_hist select from metrics_requests_stg
5. Execute table statistics in metrics_requests_hist
6. [**metrics_requests collection table clean-up**] Delete from metrics_requests
7. [**metrics_resources_usage collection table clean-up**] Delete from metrics_resources_usage
8. Update METRICS_JOB_DETAILS set 'COMPLETE" status

A key aspect of the KPImetrics is the ability to process the collection data [out-of-the-box DV metrics tables]. It is through the navtive database SQL scripts that the collection data is pruned, filtered and augmented using stage tables before it is inserted into the history tables. The use of stage tables is more efficient for processing collection data in that all

insert/update/delete operations are performed on the stage tables. Only inserts are performed on the data mart history tables.

The diagram below shows the collection data flowing from the out-of-the-box DV metrics tables through the stage tables and finally into the KPImetrics history tables. Along the way it uses various supporting tables to filter and augment the data.



Every two hours [configurable], a DV trigger wakes up and executes a native database SQL script called P_METRICS_ALL_TABLES. This script performs many operations defined later in the sections “Physical Oracle Data Transfer Script” and “Physical SQL Server Data Transfer Script”. In general, the primary purpose of the script is to transfer the collection data to the history tables and apply some business rules as related to each collection table:

metrics_sessions

1. Use stage “metrics_sessions_stg” for all inserts, updates and deletes.
2. Insert collection data from metrics_sessions into metrics_sessions_stg and augment (join) with data correlated in METRICS_ALL_USERS.
3. Update stage metrics_sessions_stg and augment (join) with data correlated in METRICS_LDAP_PERSON. The historical records of the user must be retained because METRICS_LDAP_PERSON will only contain the current user information.
4. Insert from stage metrics_sessions_stg into history metrics_sessions_hist.
5. Delete the collection rows and stage rows.

metrics_resources_usage

1. Use stage “metrics_resources_usage_stg” for all inserts, updates and deletes.
2. Prune rows from collection metrics_resources_usage by executing dynamic delete row filters based on the METRICS_JOB_FILTERS for each DV environment.
3. Insert collection data from metrics_resources_usage where (user, domain, resourcekind) not in METRICS_JOB_FILTERS into metrics_resources_usage_stg and augment (join) with data correlated in METRICS_ALL_USERS and METRICS_ALL_RESOURCES.
4. Update stage metrics_resources_usage_stg and augment (join) with data correlated in METRICS_LDAP_PERSON. The historical records of the user must be retained because METRICS_LDAP_PERSON will only contain the current user information.
5. Insert from stage metrics_resources_usage_stg into history metrics_resoures_usage_hist.
6. Delete the collection rows and stage rows.

metrics_requests

1. Use stage “metrics_requests_stg” for all inserts, updates and deletes.
2. Prune rows from collection metrics_requests where no correlating row is found in collection metrics_resources_usage.
3. Insert collection data from metrics_requests where exist in metrics_resources_usage into metrics_requests_stg and augment (join) with data correlated in METRICS_ALL_USERS.
4. Update stage metrics_requests_stg and augment (join) with data correlated in METRICS_LDAP_PERSON. The historical records of the user must be retained because METRICS_LDAP_PERSON will only contain the current user information.
5. Update stage metrics_requests_stg and augment (join) with the resourcekind and dataservicename correlated from stage metrics_resources_usage_stg.
6. Insert from stage metrics_requests_stg into history metrics_requests_hist.
7. Delete the collection rows and stage rows.