# TIBCO®

# KPI Metrics Configuration Guide

## 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 |

## Revision History

| Version | Date | Author | Comments |
|---|---|---|---|
| 1.0 | Oct 2012 | Manny Bhatia | Initial revision |
| 1.4 | Nov 2012 | Matthew Lee | Minor updates to deployment language to clarify steps |
| 1.5 | Nov 2012 | Matthew Lee | Revised setup and configuration steps for clarification |
| 1.6 | Sep 2013 | Matthew Lee | Revised document for KPImetrics version 2.0 |
| 1.7 | Jan 2014 | Matthew Lee | Migrated to Cisco documentation standards |
| 1.8 | Aug 2017 | Mike Tinius | Upgraded to work with Data Virtualization metrics with Postgres.  Add-on to DV metrics capability. |
| 1.9 – 2017.4 | Dec 2017 | Mike Tinius | Add additional capabilities and migrated to Tibco documentation standards |
| 2018.1 | March 2018 | Mike Tinius | Optimization of Oracle and SQL Server views to achieve better push-down.  Improved installation scripts. |
| 2018.101 | March 30 2018 | Mike Tinius | Fixed an issue with the installation scripts. Added KPImetrics_worksheet.xlsx. |

## Related Documents

| Name | Version |
|---|---|
| How To Use Utilities.pdf | 2018Q1 |

## Supported Versions

| Name | Version |
|---|---|
| TIBCO® Data Virtualization | 7.0.4 or later |
| AS Assets Utilities open source | 2018Q1 or later |

# Table of Contents

# 1 Introduction

**Purpose**

The purpose of this document is to provide guidance on how install, monitor and use 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

  o Cisco Data Virtualization (DV)

  o Composite Information Server (CIS)

# 2 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.

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

- Session information

    o Client session information but does not contain user unless correlated with resource usage.

    o Client throughput and duration.

- Request information

    o Captures the request from a client such as the request (SQL statement), duration, rows affected, max memory, and max disk usage.

- Resource usage information:

    o Resource accessed with start/end time and user.

    o Data source path and type and resource path and type.

    o 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:

■ metrics_requests
■ metrics_resources_usage
■ metrics_sessions

**Singe-node Environments: [DEV and SIT]**     **Cluster Environments: UAT and PROD**
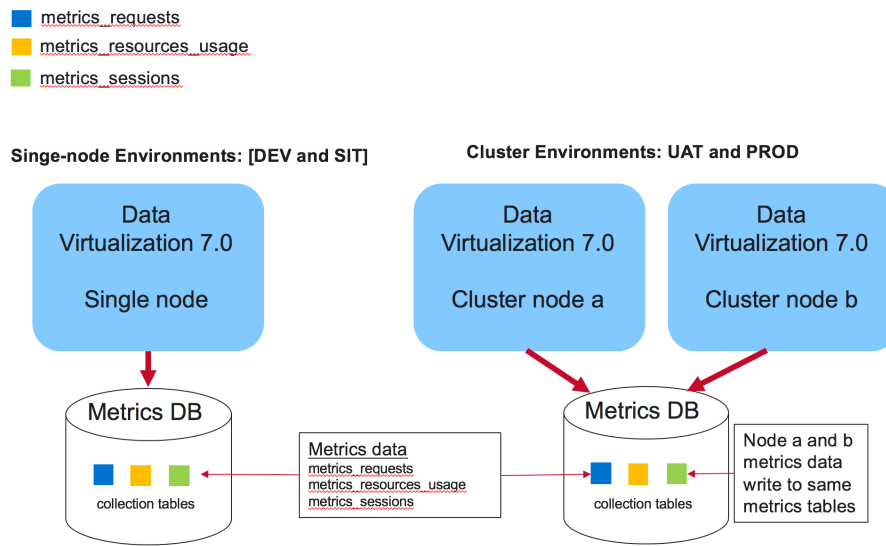


The biggest advantage for using **ASAssets KPImetrics** in addition to the DV metrics is as follows:

- Reporting strategy

  o Consider the DV metrics out-of-the-box as collection or staging 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/staging tables and data mart tables.

  o Why Collection/Staging 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

    ▪ Ability to create no logging tables and tablespace to minimize REDO and TEMP space issues when using Oracle

  o 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

    o 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 <u>not</u> collected by DV metrics allowing performance, usage and capacity planning reports may be generated on demand.

    o 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

    o Server resource utilization

        ▪ Memory and CPU Utilization of physical server

        ▪ Overall throughput of data [Memory, IO, Disk, Caches, Datasource]

        ▪ Number of sessions and requests executed on DV server

    o Developer resource utilization

        ▪ Resource ownership for a developer

        ▪ Reusability/Dependency of resources

        ▪ Identification of orphaned resources. (published but not used)

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.
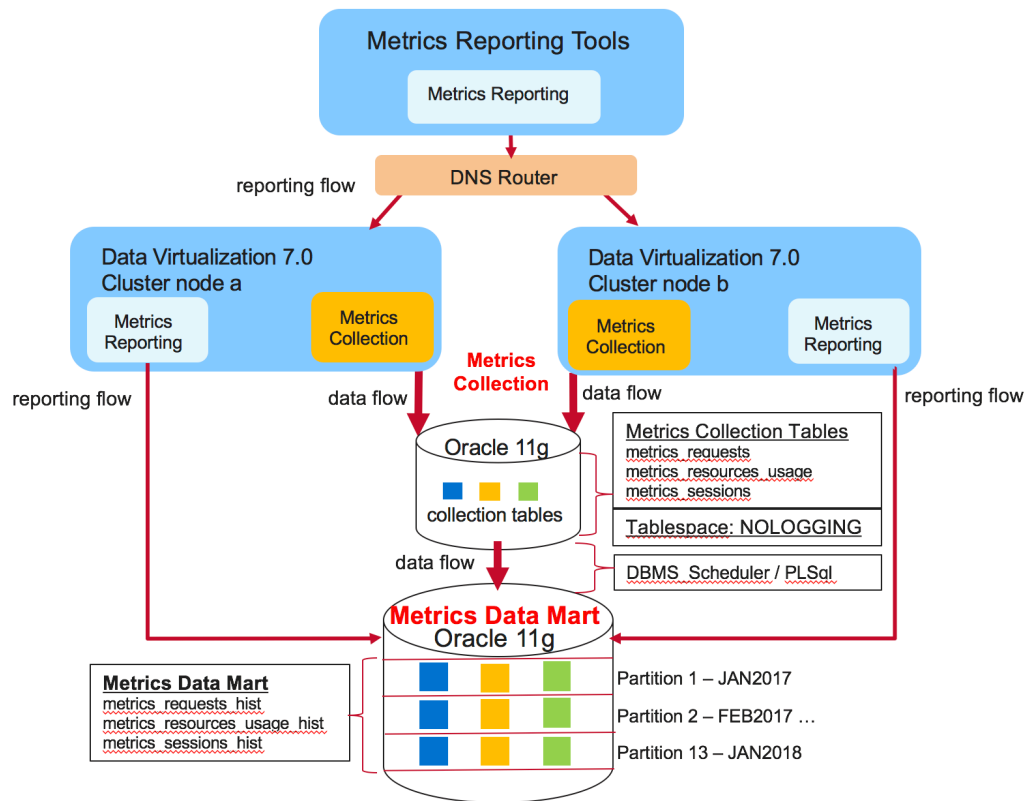
*Reporting Flow*

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.

# 3    Requirements

The following requirements and pre-requisites must be met:

- **Minimum** of **Data Virtualization (DV) 7.0.4.00 HF04** must be used as there have been several DV Metrics bug fixes to make Metrics usable.

- TIBCO Open Source **ASAssets Utilities_2018Q1.zip** is required as a baseline.  Follow the instructions within the zip file for installation.

- Database Schema for each DV environment to store the Metrics and KPImetrics data for a period of time specified by the user.  Suggestions: 4 months in lower-level environments (LLE) and 13 months in production (PROD) environments.

    o  [Oracle 11g or 12c, SQL Server 2012 or 2014]

    o  Recommended database=Oracle

        ▪  Recommended due to better push-down capabilities

        ▪  Straight-forward partitioning support

    o  Recommended schema (user) or database name=CIS_KPI

        ▪  Must be granted privileges to create/drop tables, create/drop partitions, indexes, views, sequences, PLSQL, and insert/update/delete data.

        ▪  GRANT CREATE SESSION, CREATE VIEW, ALTER SESSION, CREATE SEQUENCE, CREATE SYNONYM, CREATE DATABASE LINK, RESOURCE TO CIS_KPI;

    o  Recommended tablespaces or filegroups

        ▪  Tablespace/Filegroup name=METRICS_DATA

            •  This will contain the DV metrics collection tables [metrics_sessions, metrics_requests, metrics_resources_usage]

            •  Configured to be a NOLOGGING tablespace so that it is more efficient for providing insert/delete operations every 2 hours without impacting performance.  Rows will be transferred via to the history tables which are stored in the tablespace METRICS_DATA.

- Tablespace/Filegroup name=METRICS_DATA_HIST

  - This will contain the KPImetrics tables.

  - Configured with logging and large enough to hold 100 GB and grow as needed.

- Tablespace/Filegroup name=METRICS_DATA_IDX

  - This will contain the KPImetrics indexes.

  - Configured with logging and large enough to hold 100 GB and grow as needed.

# 4   Installation and Configuration

## Supported Database Platforms

The majority of metrics discussed in the previous sections are generated using custom aggregation procedures. Because DV does not retain the system metrics data needed to generate KPImetrics data long enough for historical reporting, the KPImetrics module must store this cached data to a dedicated database in order to retain the generated results.

The KPImetrics module supports the following database platforms at this time as incremental caching targets.

- Oracle 11g or later (recommended due to better push-down capabilities)

- SQL Server 2012 or 2014

Support for additional platforms would require customization of the KPImetrics module by a professional services consultant. Please contact TIBCO's professional service group for details.

*Please note that it is strongly recommended that the database chosen to cache KPImetrics data have case sensitivity and ignore trailing space settings that match your DV server to maximize query pushdowns in order to minimize the amount of additional load the KPImetrics module adds to your DV environment.*

## Installing KPImetrics

### Download and Import the KPImetrics components to your DV instance

Deploy the KPImetrics components to your DV instance in order to use the KPImetrics module.

1.  **Download** the ASAssets Utilities and KPImetrics from the TIBCO Open Source GIT site
    a.  Utilities: Utilities_2018Q101.zip
        i.  Follow the Utilities documentation "How To Use Utilities.pdf" for installation. Do this first.
        ii.  Utilities_2018Q1.car
    b.  KPImetrics: KPImetrics_2018Q101.zip
        i.  KPImetrics_2018Q101_installation.car
        ii.  KPImetrics_2018Q101.car
        iii.  KPImetrics Configuration Guide v1.11.pdf
        iv.  KPImetrics_worksheet.xlsx
        v.  KPImetrics_scripts directory

Complete the following steps to configure the KPImetrics components

1. **Login to DV Studio as "admin"**

   a. All configuration operations should be completed as DV "admin".

2. **Import KPImetrics Installation CAR File**:

   a. In the Studio left resource panel tree, right click on the root folder (/) icon and select Import.

   b. Import the file **KPImetrics_YYYYQn_installation.car** with the overwrite checkbox enabled. The folders /shared/ASAssets/KPImetrics_installation and should appear after the import completes.

3. **Execute Pre-Installation Script**:

   a. Execute "**1_Pre_Installation**" and provide parameters:

      i. Location: /shared/ASAssets/KPImetrics_installation/1_Pre_Installation

      ii. IN **metrics_app_id_password** - DV password for the user/owner of KPImetrics source code which is "metrics_app_id".

      iii. IN **car_file_os_full_path** - Full path to the car file archive in the OS file system.  If null the import is skipped.   If left null, the KPImetrics_YYYYQn.car will need to be imported manually.

   b. Information Only Section

      i. This script performs the following operations:

      ii. Create the published datasource "*ASAssets*" if it does not already exist.

      iii. Create the "KPImetrics" catalog for ASAssets data source.

      iv. Create a DV user called "metrics_app_id" in the "composite" domain. This way to can determine the process id that is executing requests and filter these requests out of the metrics history tables if you choose, using a strategy to be discussed later.

      v. Import KPImetrics_YYYYQn.car if path is provided.

4. **Import KPImetrics CAR File**: [Optional if not done in step 2.]

   a. **Bypass this step if you provided the path in Step 2**. above and the car file was successfully imported.  Otherwise proceed with the instructions below.

   b. In the Studio left resource panel tree, right click on the root folder (/) icon and select Import.

   c. Import the file **KPImetrics_YYYYQn.car** with the overwrite checkbox enabled. The folders /shared/ASAssets/KPImetrics and /services/databases/ASAssets/KPImetrics appear after the import completes.

**Configuration Overview**

The following is an overview of resources to be configured prior to post-installation execution.

Recommended approach:  Use **KPImetrics_worksheet.xlsx** as a way to gather the required information for this section.

1. Configure KPImetrics Data Source

2. Configure DV Email

3. Configure Common Values Procedure

4. Configure LDAP Data Source

5. Configure Metrics Job Lookup Tables

6. Configure KPImetrics Triggers

**[1.] Configure the KPImetrics data source**

The KPImetrics module makes use of several custom tables to store logging and metrics data. You must configure a data source connection in order to view KPImetrics data.

1. **Configure KPImetrics datasource**:

   a. Locate and configure the data source for your KPImetrics database.  Enable the data source that is required.
      i. **Oracle**:  **Recommended schema (user)=CIS_KPI**
         /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_oracle

      ii. **SQL Server**: **Recommended database name=CIS_KPI**
         /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_sqlserver

   b. Disable the data sources not being used.
   c. Test the connection to make sure it is working properly.
   d. If you used a schema name or database name different than CIS_KPI, the post installation script will automatically take care of this.

**[2.] Configure DV Email**

The KPImetrics module uses sends alerts and notifications.

2. **Configure DV email:**

   a. Select Administration → Configuration → Server → Configuration → E-mail
      i. From Address
      ii. SMTP Host Name
      iii. If required, provide user name and password

**[3.] Configure Common Values**

The KPImetrics module uses several constant values that are set in the procedure /shared/ASAssets/KPImetrics/Configuration/**commonValues**. You will need to update some of these constants with values for your environment to ensure that KPImetrics functions correctly. Update the following minimum values for operation.

1. **Configure Common Values:**

    a. Open the procedure /shared/ASAssets/KPImetrics/Configuration/commonValues and modify the following properties:

    b. **cisServerNickname** – This uses the Utilities: /environment/getEnvName() procedure.  Configure getEnvName() with the DV server nickname for the instance of DV.  E.g. DEV1, SIT1, UAT1, PROD1.  Alternatively, a static string could be used as well.  This is used in email notifications to alert you of issues.  You determine what your environment nicknames are.  Be consistent.  _Each DV server environment must be unique_.  For a DV cluster, each node in the cluster will have the same nickname as it describes the environment and not the node.

    c. **sendTo** – Provide the comma separated list of email address to send notifications to.  This setting is used as a fall back if there are no METRICS_EVENT_REGISTRATION rows configured and, in the event, that the database is completely down.  Note: You must configure email in DV.

    d. **CpuCheckerCommandPath** – Determine which script format to use based on your installation environement.
        i. The following section stores CPU and memory checker default values for WINDOWS and UNIX.  The following are the command line execution statements:
        ii. CPU Utilization - processor time percentage
        iii. Windows 7 or 10: 'powershell.exe -file '||getCisHome()|| '\bin\KPImetricsCpuUtilization.ps1'
        iv. UNIX (Linux 6): getCisHome()||'/bin/KPImetricsTopCommandGrepCpu_linux6.sh'
        v. UNIX (Linux 7): getCisHome()||'/bin/KPImetricsTopCommandGrepCpu_linux7.sh'

    e. **memoryCheckerCommandPath** – Determine which script format to use based on your installation environement.
        i. Memory Utilization - Memory used and memory available
        ii. Windows 7 or 10: 'powershell.exe -file '||getCisHome()|| '\bin\KPImetricsMemUtilization.ps1'
        iii. UNIX (Linux 6): getCisHome()||'/bin/KPImetricsFreeMemCommand_linux6.sh'
        iv. UNIX (Linux 7): getCisHome()||'/bin/KPImetricsFreeMemCommand_linux7.sh'

    f. **dataSourceName** – Provide the name of the data source used to store KPImetrics data. Valid values are KPI_oracle or KPI_sqlserver.

    g. **Data source schema path** – provide the schema path for the appropriate datasource type being used.  If you used a different schema or database

name than the recommended CIS_KPI you will modify the appropriate /schema or /catalog/schema path below for your database type.

  i. dataSourceSchemaPath_oracle – e.g. /CIS_KPI
  ii. dataSourceSchemaPath_ sqlserver – e.g. /CIS_KPI/dbo

h. **collectionTablespaceName** – Identifies the Metrics collection tablespace name which will contain the DV out-of-the-box metrics tables.  To use the default tablespace or no tablespace set this value to NULL.  You should have already created a tablespace (with no logging if Oracle) or a filegroup if SQL Server.

i. **historyTablespaceName** – Identifies the Metrics history tablespace name which will contain all of the history and reporting tables.  To use the default tablespace or no tablespace set this value to NULL.  You should have already created a tablespace or a filegroup if SQL Server.

j. **indexTablespaceName** – Identifies the Index tablespace name which will contain the indexes for the history tables.  To use the default tablespace or no tablespace set this value to NULL.  You should have already created a tablespace or a filegroup if SQL Server.

k. **partitionNumber** – Identifies the number of table partitions to use for the metrics history tables.  The default is 0 indicating no partitioning will be used.  It is "highly" recommended to partition the three history tables as it will make queries more performant and make it easier to drop a partition to purge data rather than executing delete statements.  Recommendation: Always add 1 additional month to your standard data retention policy to allow for dropping a partition of data.

  i. (LLE) lower-level env:        policy=3. Set to 4 months of data
  ii. (PROD) production-level env:   policy=12. Set to 13 months of data.

l. **partitionStartDate** – Identifies the starting date for the partitioning scheme in the format 'YYYY-MM-DD'.  If null then no partitioning date is used.  If partitionNumber is > 0 and partitionStartDate is null an error is thrown.  If the partition format is not correct and it cannot be cast to a DATE then an error is thrown.  This should be the first of the month in which metrics are turned on and capturing data.  If metrics have been previously turned on and have been capturing data for a while then use the first of the month of the date they were turned on.  Execute the following query against the current metrics table to determine the date:  select min(starttime) starttime from metrics_requests

m. **dataTransferBatchMax** – Identifies the batch maximum number for the P_METRICS_ALL_TABLES data transfer procedure.  After the metrics_requests collection table is transfered into the metrics_requests_hist table, an update routine executes to correlate the resourcekind and dataservicename from the metrics_resources_usage_hist with the metrics_requests_hist table.  This routine commits rows from the metrics_requests_stg table for the batch number specified here.  Note: This is currently only implemented for Oracle and is set to 50000.

**[4.] Configure LDAP Data Source**

The KPImetrics module is designed to retrieve user data from an LDAP directory server in order to provide additional detail on which users are making use of a monitored DV environment. You may configure the LDAP data source provided with the KPImetrics module to connect to your corporate LDAP directory server.  The essence of many of the queries is based on LDAP data and LDAP users who connect to DV to execute queries.  If you do not have LDAP, then consider the options below.

1. **Configure the LDAP data source**
   a. **Option 1** – Configure the KPImetrics LDAP datasource to connect to your local ldap:  /shared/ASAssets/KPImetrics/Physical/Metadata/LDAP
      i. Proceed to #2 below to map LDAP_PERSON to the LDAP datasource.
   b. **Option 2** – Configure your own LDAP datasource.
      i. Proceed to #2 below to map LDAP_PERSON to your datasource.
   c. **Option 3** – Configure a pseudo LDAP to satisfy the inner workings of KPImetrics.
      i. Proceed directly to #2 below and modify the existing UNION statement with composite users.  There is no mapping to LDAP in this scenario.
2. **Modify the LDAP_PERSON view**
   /shared/ASAssets/KPImetrics/Physical/Metadata/System/LDAP_PERSON
   Correctly map to your LDAP directory structure.

   a. For Option 1 and 2, comment out the entire default UNION section of 'user1', 'user2' and 'user3' at the bottom of the view.
      ii. Uncomment the top section which is the LDAP implementation section.
      iii. Modify the source location "FROM" clause as required: FROM /shared/ASAssets/KPImetrics/Physical/Metadata/LDAP/organizationalPerson
   b. For Option 3, modify the existing UNION and modify 'user1', 'user2', and 'user3' with your own composite users.
   c. Guidelines:
      iv. Keep the alias column names the same.
      v. The physical column names may be different based on your LDAP data source.
      vi. If a column does not exist then simply do a CAST(null as VARCHAR(…)) columnName.
      vii. Keep the number of columns the same.
      viii. Keep the datatype and length the same.
      ix. Implement a where clause as required based on your LDAP data source.  E.g. WHERE userid IS NOT NULL AND employeeStatus IS NOT NULL and  employeeStatus = 'A' – active employees only
   d. Key fields to map include the following:
      x. cn (lower case userid field)

          xi.   objectClass

          xii.   displayName

          xiii.   name

          xiv.   objectGUID (user id field)

          xv.   sn (surname)

          xvi.   givenName

          xvii.   employeeNumber

          xviii.   mail

          xix.   baseDN

          xx.   relativeDN

e.   Nice to have fields to map include the following:

          xxi.   description

          xxii.   telephoneNumber

          xxiii.   c (country)

          xxiv.   l (city)

          xxv.   st (state)

          xxvi.   street (street address)

          xxvii.   ou

          xxviii.   title

          xxix.   postalAddress

          xxx.   postalCode (zip code)

          xxxi.   postOfficeBox

          xxxii.   physicalDeliveryOfficeName

          xxxiii.   initials

          xxxiv.   employeeType

          xxxv.   manager

          xxxvi.   homePhone

          xxxvii.   mobile

          xxxviii.   pager

## [5.] Configure Metrics Job Lookup Tables

The KPImetrics contains several lookup tables that need to be pre-populated prior to the installation script running.  The tables include:

- METRICS_JOB_ENVIRONMENTS
- METRICS_JOB_FILTERS
- METRICS_EVENT_REGISTRATION

1. **Configure 08_pqInsert_KPI_Tables_METRICS_JOB_tables**

a. Edit /shared/ASAssets/KPImetrics/Physical/Metadata/DDL/Common/

08_pqInsert_KPI_Tables_METRICS_JOB_tables

      i.   Configure the ***METRICS_JOB_ENVIRONMENTS***

1. Add a unique row for each environment in your pipeline. Remove any example rows that you are not relevant.

2. Example: DEV1, SIT1, UAT, PROD

ii. Configure the **METRICS_JOB_FILTERS** – The job filters are used to filter out rows by the data transfer routine when transferring from the collection tables to the history tables. The objective is to identify potentially high-use, non-user defined requests that are of little or no value to the ultimate KPImetrics reporting. *Since this feature is not built into DV metrics, it is "highly" recommended to filter out admin and other user accounts associated with deployment or non-query type of activity in DV.*

1. Example 1 would be for each environment:

ENV_TYPE TABLE_NAME            USER      DOMAIN RESOURCE_KIND
'UAT',       'metrics_resources_usage', 'admin', 'composite', 'system'

2. Example 2 would be any of the KPImetrics functionality that produces their own events. The assumption is that the KPImetrics folder resources were configured for resource ownership by metrics_app_id during installation:

ENV_TYPE TABLE_NAME            USER      DOMAIN RESOURCE_KIND
'UAT', 'metrics_resources_usage', 'metrics_app_id', 'composite', 'system'

3. Example 3 would be a deployment process that produces an inordinately high volume of events. Assume you have a deployment user called "deploy_app_id" that runs a deployment process but you don't want to log those events:

ENV_TYPE TABLE_NAME            USER      DOMAIN RESOURCE_KIND
'UAT', 'metrics_resources_usage', 'deploy_app_id', 'composite', 'system'

4. Add a unique row for each environment, user, domain and resourcekind combination.

b. Post-installation maintenance

i. This script can be executed by itself post-installation by simply executing it to reload what is configured.

2. **Configure *09_pqInsert_KPI_Tables_METRICS_EVENT_REGISTRATION***

a. Edit /shared/ASAssets/KPImetrics/Physical/Metadata/DDL/Common/ 09_pqInsert_KPI_Tables_METRICS_EVENT_REGISTRATION

a. **Requirement**:   Prior to execution of this script LDAP data must first be configured and loaded.  The post-installation script will automatically cache LDAP_PERSON as long as it is configured and caching is turned on server-wide.

b. Configure the METRICS_EVENT_REGISTRATION

1. This table contains the event registrations for sending emails based on the subscriptions to various events.  This procedure is used for either initial load or maintenance.  If a row already exists, it does not update it.  It simply bypasses it.  This means that you can run this procedure as many times as you want and not impact existing rows.  It does not delete or unsubscribe requester events.  To delete a subscription, invoke pMetricsEventRegistrationUnsubscribe().  A subscription in the METRICS_EVENT_REGISTRATION table consists of a unique record for the combination of SUBSCRIBER_EMAIL, GROUP_NAME, ENVIRONMENT_TYPE, EVENT_TYPE and REQUESTER_EMAIL.

2. Edit a row to provide the subscription information.  At a minimum, add a group subscriber email for the DV administration group for each event.  If you don't have a group alias, then choose the DV administrator's email to receive alerts.  A row includes the following:

   a. REQUESTER_EMAIL [PK] – Primary requester email.

   b. SUBSCRIBER_EMAIL [PK] – userid email or a group email alias.  Who the email alert will be sent to.

   c. GROUP_NAME [PK] – Group name subscribing to. When an alert occurs for a user the groups will be checked and cross-referenced with this registered group.  The group [all] is a composite group and a catch-all for any user belonging to this composite group.

   d. ENVIRONMENT_TYPE [PK] – Register for all environments [ALL] or a certain environment type [DEV1, CIT1, SIT1, UAT, TT, PROD]

   e. EVENT_TYPE [PK] – [LONG_RUNNING, EXCEEDED_MEMORY, INACTIVITY, WORKFLOW_FAILURE, DBMS_SCHEDULER_ERROR,

PURGE_HISTORY]

      f.  EXCLUDE_TEXT – A comma separate list of text that when found will signal exclusion and the email will not be sent. This is a way of filtering out emails based on text.

  c.  Post-installation maintenance

      i.  This script can be executed by itself post-installation by simply executing it and answering "Y" to the parameter to delete the rows and reload what is configured.

## [6.] Configure KPImetrics Triggers

The KPImetrics module uses a series of triggers to cache various tables of information. The different triggers provide flexibility to turn on and off processing as required. If certain functionality and data is not required, the trigger can be turned off saving on database space and DV processing.

    **1.  Configure default triggers for your use case**

      a.  Open/Edit the resource /shared/ASAssets/KPImetrics/Configuration/**defaultTriggersToEnable**

      b.  Only modify the ON/OFF settings for each trigger. Leave all other settings alone.

          i.  Refer to the following sections for details on each trigger:

              1.  [Metadata System Triggers and Load Scripts](#)

              2.  [Physical Oracle Triggers and Scripts](#)

              3.  [Physical SQL Server Triggers and Scripts](#)

      c.  The current triggers defaulted to OFF are as follows:

          i.  **kpimetricsTrig_00_CheckMetricsActivityDebug** – Only turn this on if you suspect that DV metrics is not working properly and you want to debug the DV metrics every hour.

          ii.  **kpimetricsTrig_12_Cache_METRICS_SQL_REQUEST_REPROCESS** – Only turn this on if you get a code update from the Open Source site and there were changes to the SQL Parser code.

          iii.  **kpimetricsTrig_18_CheckExceedMemoryPercentRequests** – You may choose to keep this off in lower-level environments but turn it on in PROD environments.

          iv.  **kpimetricsTrig_19_CheckLongRunningRequests** – You may choose to keep this off in lower-level environments but turn it on in PROD environments.

d. The current triggers defaulted to ON that you may wish to evaluate:

    i. **kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST** – This trigger is defaulted to ON.  If you do not wish to perform SQL parsing on the request description SQL statement to parse out the table and column resources used in the SQL then turn this trigger off.  There is quite a bit of overhead associated with this trigger.

e. **Trigger Category – Essential**

    i. Provides baseline data for the data transfer and other processes

        1. kpimetricsTrig_01_Cache_ALL_RESOURCES

        2. kpimetricsTrig_02_Cache_ALL_USERS

        3. kpimetricsTrig_03_Cache_LDAP_PERSON

    ii. Used to transfer data from collection tables into history tables

        1. kpimetricsTrig_30_DBMSScheduler_KPI_oracle

        2. kpimetricsTrig_31_DBMSSchedulerError_KPI_sqlserver

    iii. Used to check for errors with the data transfer

        1. kpimetricsTrig_31_DBMSSchedulerError_KPI_oracle

        2. kpimetricsTrig_31_DBMSSchedulerError_KPI_sqlserver

    iv. Used to perform partition management on the history tables

        1. kpimetricsTrig_32_DBMSPartitionManager_KPI_oracle

        2. kpimetricsTrig_32_DBMSPartitionManager_KPI_sqlserver

    v. Provides clean-up/purge capability based on define schedule

        1. kpimetricsTrig_16_PurgeHistoryData

    vi. Provides monitoring capabilities

        1. kpimetricsTrig_14_CheckCISWorkflowStatusFail

        2. kpimetricsTrig_15_CheckMetricsActivity

f. **Trigger Category – 2nd level processing (lite-weight)**

    i. A series of value-added metrics that are an addition to the DV out-of-the-box metrics that are lite-weight in terms of processing.  DV does not incur much overhead when these triggers execute.

        1. kpimetricsTrig_04_Cache_CIS_SYSTEM_RESOURCES

        2. kpimetricsTrig_05_Cache_CPU_MEMORY_CHECKER

        3. kpimetricsTrig_06_Cache_LOG_DISK

        4. kpimetricsTrig_07_Cache_LOG_IO

        5. kpimetricsTrig_08_Cache_LOG_MEMORY

        6. kpimetricsTrig_12_Cache_SYS_CACHES

        7. kpimetricsTrig_13_Cache_SYS_DATASOURCES

8. kpimetricsTrig_17_CheckExceedMemoryPercentRequests

9. kpimetricsTrig_18_CheckLongRunningRequests

g. **Trigger Category – 3rd level processing (heavy-weight)**

i. A series of value-added metrics that are an addition to the DV out-of-the-box metrics

1. kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST

2. kpimetricsTrig_19_AllCustom_AccessByUserOverTime

3. kpimetricsTrig_20_AllCustom_ActiveResourcesOverPeriodOfTime

4. kpimetricsTrig_21_AllCustom_ResourceCount_Details

5. kpimetricsTrig_22_AllCustom_ResourceCount_Total

## Execute Post-Installation Script

The KPImetrics module provides an automated script to complete the installation.  The "Information Only Section" below will describe in detail what the script is going to execute.  When the installation is complete and there are no red/impacted resources then turn on the triggers to begin processing KPI metrics data.

**Execute Installation Script**

Perform the post-installation configuration.

1. **Execute Post-Installation Script**:

   a. WARNING:  If you are executing an installation into an existing KPImetrics environment, this script **WILL DROP** existing tables and data.  Do not use this script for upgrading an environment.  We recommend using TIBCO Professional Services for doing upgrades.

   b. Execute "**2_Post_Installation**" and provide parameters:

      i. Location: /shared/ASAssets/KPImetrics_installation/2_Post_Installation

      ii. IN inDebugSecondary – Debug Secondary provides a deep debug with detailed debug output.  Y=deep debug.   N=cursory, high-level debug.  N is recommended unless there are issues.

      iii. IN **performInstallationAction** – Y=perform the installation which will drop and recreate KPImetrics tables/sequences/procedures. N=Do nothing.

      iv. IN **destroyCIS_cache_status_Tables** – Y=destroy and recreate cache_status, cache_tracking.  N=do not destroy if they exist.

      v. IN **destroyCIS_metrics_collection_Tables** – Y=destroy and recreate DV metrics collection tables including metrics_requests, metrics_resources_usage, and metrics_sessions.  N=do not destroy if

they exist.

vi.  IN **forceOverwrite** – Force overwriting the CPU and Memory checker Windows/UNIX scripts.  Y=force overwrite.  N=do not force overwrite.

**Deploy CPU and Memory Checker shell scripts (Windows and UNIX)**

During post-installation, it was attempted to write the scripts to the $CIS_HOME/bin directory on either Windows or UNIX depending on your operating system.

a.  If <u>no error was returned</u> then all scripts were installed correctly into $CIS_HOME/bin and are able to be executed.  Continue to the next step.

b.  If the <u>error message</u> "*23. INSTALL KPImetrics CpuAndMemoryChecker scripts manually.*" was received during post-installation, then you will need to install the scripts manually and set the file permissions accordingly especially for UNIX such as [rwxr-xr-x]. The recommend location to deploy the scripts is $CIS_HOME/bin so that they can be executed by the user account that DV is running under. Take note of where the scripts have been deployed, you will need to provide the path to the scripts when configuring the KPImetrics "commonValues" script. The datasource "/shared/ASAssets/KPImetrics/Physical/Metadata/CPUAndMemChecker" is used to execute the scripts. The following details the scripts:

   a.  Windows Powershell
      i.  KPImetricsCpuUtilization.ps1
      ii.  KPImetricsMemUtilization.ps1
   b.  Linux6_scripts
      i.  KPImetricsFreeMemCommand_linux6.sh
      ii.  KPImetricTopCommandGrepCpu_linux6.sh
   c.  Linux7_scripts
      i.  KPImetricsCpuFormat_linux7
      ii.  KPImetricsFreeMemCommand_linux7.sh
      iii.  KPImetricTopCommandGrepCpu_linux7.sh

**Enable Triggers**

Enabling triggers starts the processing of KPI metrics data. The KPImetrics module makes use of pure database tables in order to retain DV metrics for a longer period than supported by the base DV logging functionality.

*Please note that executing this procedure will cache the minimum required tables: METRICS_ALL_RESOURCES, METRICS_ALL_USERS and METRICS_LDAP_PERSON. Those 3 tables must be completed before any of the other triggers are allowed to run.*

2.  **Enable Triggers**:

   a.  Execute the **updateTriggers** procedure

  i. Location: /shared/ASAssets/KPImetrics/Configuration/**updateTriggers**

  ii. *enable* – 0=disable, 1=enable, 2=display trigger list

    1. Select 1 to enable all configured triggers.

  iii. *includeList* – Comma-separated list of trigger numbers to include in the (enable/disable) action.  Leave null if the "defaultTriggersToEnable" are configured as desired.

  iv. *excludeList* – Comma-separated list of trigger numbers to exclude in the (enable/disable) action.  The excludeList overrides includeList.  Leave null if the "defaultTriggersToEnable" are configured as desired.

## Configure DV Out-of-the-box Metrics

Go to the "Configure DV Metrics" section.

## Information Only Section

**This section provides a background on what gets executed by the post-installation script.**

*Script Requirements:*
1. This script must be executed by the user admin or someone who has admin privileges
2. The KPImetrics CAR file has been imported either manually or with 1_Pre_Installation.

*Steps to be performed after the KPImetrics CAR file is imported:*
1. Update impacted resources.
2. Set resource privileges
3. Enable and test the KPImetrics cache datasource
4. Validate and Create cache_status and cache_tracking if not exist
5. Validate and Create KPImetrics collection tables exist
6. Create KPImetrics cache tables, sequences and procedures [This will drop existing]
7. Introspect / Reintrospect the KPImetrics cache datasource
8. Change resource ownership
9. Update the system cache datasource and tables
10. Rebind physical database type tables to the datasource schema
11. Rebind the physical abstraction folder "/Physical/Abstraction" to the correct physical datasource folder "/Physical/KPI_[oracle|sqlserver]"
12. Remove CIS_KPI folder if not needed
13. Update impacted resources
14. Update all cache tables
15. Refresh the LDAP_PERSON view
16. Load the METRICS_JOBS table
17. Load the METRICS_EVENT_REGISTRATION table

18. Install the CpuAndMemCheckerCjp scripts into the file system for either Windows or UNIX.
19. Update the CPUAndMemChecker data source url and reintrospect it
20. Test the /System/CPU_MEMORY_CHECKER view

How the scripts work:

The KPImetrics module requires several tables in the KPImetrics data source database in order to store metrics data for reporting. You must create these storage tables using the provided DDL in order for the KPImetrics module to function correctly.

If you choose to create the tables from within Studio, execute the procedures under /shared/ASAssets/KPImetrics/Physical/Metadata/DDL for your data source type. Proceed to the following section associated for your database type.

Procedure Parameters:

**IN displayDDL –** Y=Display the DDL, N=Do not display the DDL.

**IN executeDDL** – Y=execute the DDL, N=Do not execute the DDL. If you choose to execute the DDL externally, you will need to execute each of the 01-07 DDL procedures with the variable set as executeDDL=N and displayDDL=Y. This will output the DDL that you need to execute externally without actually performing the DDL operations. Once you have executed each procedure 01-07, you are now ready to run the DDL externally. Once you have executed the DDL proceed to the next section "*Common Configuration for all Databases*".

**IN dropIndexes**– Y=drop the indexes before creating the first. N=do not drop the indexes.

**IN dropTables** – Y=drop the tables before creating the tables. N=do not drop the tables.

**IN createTables** – Y=execute the table creation DDL, N=display the table creation DDL in the console window only.

**IN createIndexes**– Y=execute index creation DDL. N=display the index creation DDL in the console window only.

**OUT cursCombinedResult** – Provides a status on each SQL statement executed.

**OUT sqlScript** – Generates an output of the entire script which can be used for external execution.

The common 08-09 DML procedures to populate the database must be run from within DV as there is no external SQL generation for those.

**Create the KPImetrics storage tables for Oracle**

1. Do the cache_status and cache_tracking tables exist in this database?
    a. NO – Execute **01_pqCreateDrop_KPI_Tables_oracle_cache_system**
    b. YES – Bypass this step
2. Do the DV metrics collection tables exist?
    a. NO
        i. Create a tablespace called "METRICS_DATA" with NOLOGGING.
            1. Example:
            2. create tablespace METRICS_DATA nologging datafile 'C:/DV/oracle/metrics_data01.dbf' size 500m autoextend on next 50m extent management local;
        ii. Execute

            **02_pqCreateDrop_KPI_Tables_oracle_metrics_collection_staging_tables**
            1. SchemaName – derived from commonValues.dataSourceSchemaPath_oracle
            2. TablespaceName – derived from commonValues.collectionTablespaceName
        iii. Execute

            **02_pqCreateDrop_KPI_Tables_oracle_metrics_collection_tables**
            1. SchemaName – derived from commonValues.dataSourceSchemaPath_oracle
            2. TablespaceName – derived from commonValues.collectionTablespaceName
    b. YES
        i. Consider exporting the data and turning off metrics in order to create these tables with NO LOGGING option enabled in their own tablespace called "METRICS_DATA".
        ii. If this is not desired, then bypass this step.
3. Create the metrics history tables and indexes
    a. Create a tablespace called "METRICS_DATA_HIST" and "METRICS_DATA_IDX" with logging turned on.
        i. Example:

    ii.   create tablespace METRICS_DATA_HIST logging datafile 'C:/DV/oracle/metrics_data_hist01.dbf' size 500m autoextend on next 50m extent management local;

    iii.  create tablespace METRICS_DATA_IDX logging datafile 'C:/DV/oracle/metrics_data_idx01.dbf' size 500m autoextend on next 50m extent management local;

   b.  Drop and Create the metrics history tables and indexes: metrics_requests_hist, metrics_resources_usage_hist and metrics_sessions

   c.  Execute ***03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables*** with input:

    i.   partitionNumber– derived from commonValues.partitionNumber

    ii.   partitionStartDate– derived from commonValues.partitionStartDate

    iii.  SchemaName – derived from commonValues.dataSourceSchemaPath_oracle

    iv.  TablespaceName – derived from commonValues.historyTablespaceName and commonValues.indexTablespaceName.

4.  Create the metrics KPI reporting tables and indexes

   a.  Execute ***04_pqCreateDrop_KPI_Tables_oracle_kpi_tables*** with input:

    i.   SchemaName – derived from commonValues.dataSourceSchemaPath_oracle

    ii.   TablespaceName – derived from commonValues.historyTablespaceName and commonValues.indexTablespaceName.

5.  Create the metrics KPI custom application reporting tables and indexes

   a.  Execute ***05_pqCreateDrop_KPI_Tables_oracle_kpi_application_tables*** with input:

    i.   SchemaName – derived from commonValues.dataSourceSchemaPath_oracle

    ii.   TablespaceName – derived from commonValues.historyTablespaceName and commonValues.indexTablespaceName.

6.  Create the metrics KPI reporting sequence

   a.  Execute ***06_pqCreateDrop_KPI_Tables_oracle_kpi_sequence*** with input:

    i.   SchemaName – derived from commonValues.dataSourceSchemaPath_oracle

7.  Create the metrics KPI "native" PLSQL data transfer script

a. Execute *07_pqCreateDrop_KPI_Plsql_oracle_data_xfer_script* with no input:

    i. SchemaName – derived from

       commonValues.dataSourceSchemaPath_oracle

**Create the KPImetrics storage tables for SQL Serer**

1. Do the cache_status and cache_tracking tables exist in this database?

    a. NO – Execute **01_pqCreateDrop_KPI_Tables_sqlserver_cache_system**

    b. YES – Bypass this step

2. Do the DV metrics collection tables exist?

    a. NO

       i. Create a tablespace (filegroup) called "METRICS_DATA".

          1. <u>Example</u>:

          2. alter database CIS_KPI add filegroup METRICS_DATA;

          3. alter database CIS_KPI add FILE (NAME='METRICS_DATA_1', FILENAME='C:\SQLServer_FileGroup\METRICS_DATA_1') TO FILEGROUP METRICS_DATA;

       ii. Execute

       **02_pqCreateDrop_KPI_Tables_sqlserver_metrics_collection_tables**

          1. SchemaName – derived from commonValues.dataSourceSchemaPath_sqlserver

          2. TablespaceName – derived from commonValues.collectionTablespaceName

    b. YES

       i. Consider exporting the data and turning off metrics in order to create these tables with NO LOGGING option enabled in their own tablespace called "METRICS_DATA".

       ii. If this is not desired, then bypass this step.

3. Create the metrics history tables and indexes

    a. Create a tablespace (filegroup) called "METRICS_DATA_HIST" and "METRICS_DATA_IDX" with logging turned.

       i. <u>Example</u>:

       ii. alter database CIS_KPI add filegroup METRICS_DATA_HIST;

       iii. alter database CIS_KPI add FILE (NAME='METRICS_DATA_HIST_1', FILENAME='C:\SQLServer_FileGroup\METRICS_DATA_HIST_1') TO FILEGROUP METRICS_DATA_HIST;

iv. alter database CIS_KPI add filegroup METRICS_DATA_IDX;

v. alter database CIS_KPI add FILE (NAME='METRICS_DATA_IDX_1',
FILENAME='C:\SQLServer_FileGroup\METRICS_DATA_IDX_1') TO FILEGROUP
METRICS_DATA_IDX;

b. Drop and Create the metrics history tables and indexes: metrics_requests_hist,
metrics_resources_usage_hist and metrics_sessions

c. Execute *03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables* with
input:

i. partitionNumber– derived from commonValues.partitionNumber

ii. partitionStartDate– derived from commonValues.partitionStartDate

iii. SchemaName – derived from
commonValues.dataSourceSchemaPath_sqlserver

iv. TablespaceName – derived from commonValues.historyTablespaceName
and commonValues.indexTablespaceName.

4. Create the metrics KPI reporting tables and indexes

a. Execute *04_pqCreateDrop_KPI_Tables_sqlserver_kpi_tables* with input:

i. SchemaName – derived from
commonValues.dataSourceSchemaPath_sqlserver

ii. TablespaceName – derived from commonValues.historyTablespaceName
and commonValues.indexTablespaceName.

5. Create the metrics KPI custom application reporting tables and indexes

a. Execute *05_pqCreateDrop_KPI_Tables_sqlserver_kpi_application_tables* with
input:

i. SchemaName – derived from
commonValues.dataSourceSchemaPath_sqlserver

ii. TablespaceName – derived from commonValues.historyTablespaceName
and commonValues.indexTablespaceName.

6. Create the metrics KPI reporting sequence

a. Execute *06_pqCreateDrop_KPI_Tables_sqlserver_kpi_sequence* with input:

i. SchemaName – derived from
commonValues.dataSourceSchemaPath_sqlserver

7. Create the metrics KPI "native" PLSQL data transfer script

a. Execute *07_pqCreateDrop_KPI_Plsql_sqlserver_data_xfer_script* with no input:

i. SchemaName – derived from

commonValues.dataSourceSchemaPath_sqlserver

**Common Configuration for all Databases**

8. Reintrospect or Introspect the datasource
   a. Was the default CIS_KPI schema or CIS_KPI/dbo catalog/schema used?
      i. YES:  Reintrospect the KPImetrics data source to confirm that the tables are visible.
         1. Right-click on your chosen datasource "KPI_<database_type>" and select "Re-Introspect Now" and wait for it to complete.
         2. Click OK when completed.
      ii. NO: A new schema or catalog name was used.  Introspect the new schema or catalog/schema.
         1. Right-click on your chosen datasource "KPI_<database_type>" and select "Add/Remove Tables".
         2. Select your schema or catalog/schema
         3. Select all tables/procedures:
            a. P_METRICS_ALL_TABLES
            b. "cache_status" and "cache_tracking"
            c. Starting with "METRICS_..."
            d. Starting with "metrics_..."
            e. Click Next.  Click Finish.  Wait for the introspection to complete. Review the list of tables in this section:
            f. "[Metadata Data Source for KPI_<database_type>](#)"
         4. Click OK when completed.

9. Execute the update System Cache and Datasource procedure /shared/ASAssets/KPImetrics/Configuration/**updateSystemCacheDSandTables**
   a. No input is required.  It uses commonValues to determine the target data source to rebind to.  *Note*: /shared/ASAssets/KPImetrics/Configuration/commonValues must be configured prior to executing this procedure.
   b. This procedure is used to configure the datasource and cache table that the KPI_<database_type> folder views are using in the cache tab.
   c.  Steps executed:

i.  Enable the KPI_<database_type> data source located here
/shared/ASAssets/KPImetrics/Physical/Metadata/KPI_<database_type>

ii.  Update the KPI_<database_type> data source cache_status and
cache_tracking paths to use the correct schema path for the datasource
located here:
/shared/ASAssets/KPImetrics/Physical/Metadata/KPI_<database_type>

iii.  Modify the cache table datasource and cache table path located here:
/shared/ASAssets/KPImetrics/Physical/Physical/KPI_<database_type>
Iterate over
/shared/ASAssets/KPImetrics/Configuration/defaultTriggersToEnable looking for
isCache and cacheTableName

1.  If cache is configured and enabled then configure it and leave it
enabled.

2.  If cache not configured then configure it but don't enable it.

10. Execute the rebind of the physical database type folder resources
/shared/ASAssets/KPImetrics/Configuration/**rebindPhysicalDatabaseType**

a.  This procedure rebinds the
/shared/ASAssets/KPImetrics/Physical/Physical/KPI_<database_type> folder to the
configured schema and catalog path found in commonValues.

b.  No input is required.  It uses commonValues to determine the target data source to
rebind to.  *Note*: /shared/ASAssets/KPImetrics/Configuration/commonValues must
be configured prior to executing this procedure.

11. Execute the rebind procedure
/shared/ASAssets/KPImetrics/Configuration/**rebindPhysicalAbstraction** to rebind all
KPImetrics abstraction layer views to the appropriate data source.

a.  No input is required.  It uses commonValues to determine the target data source to
rebind to.  *Note*: /shared/ASAssets/KPImetrics/Configuration/commonValues must
be configured prior to executing this procedure.

b.  Based on the commonValues.dataSourceName, it will rebind the
/Physical/Abstraction views to either /Physical/KPI_oracle or
/Physical/KPI_sqlserver.

c. Additionally, there is a list of views "commonValues.viewOptimizationPathList" used by rebindPhysicalAbstraction, that is used to modify the view SQL Script to optimize for Oracle or SQL Server in order to achieve push-down of the query.

12. Remove default CIS_KPI schema/catalog if not used.

   a. If a different schema/catalog was chosen other than CIS_KPI then remove the old CIS_KPI name from your chosen datasource "KPI_<database_type>".

      i. Right-click on CIS_KPI and select delete

         1. Update Impacted Resources

   b. Potentially, there may be some resources that are red/impacted.

      ii. Session is null

      iii. Java.lang.null

13. Execute /shared/ASAssets/KPImetrics/Configuration/**updateImpacteResources**

   a. Refresh your studio once this completes and the red/impacted resources should disappear.  If they do not, then edit the ones that are still red/impacted.  Put a space anywhere in the resource and save the resource.  The act of editing and changing the resource should cause the common error "session is null" to go away.  If the error persists, perhaps it is some other issue that requires a closer look.

14. Execute *08_pqInsert_KPI_Tables_METRICS_JOB_tables* with no input

   a. Note: This same procedure is used to modify rows.  It always deletes the rows and then inserts the rows.

15. Execute *09_pqInsert_KPI_Tables_METRICS_EVENT_REGISTRATION* with input:

   a. deleteAllRows – Y=delete all rows first, N=Do not delete all rows. (default).

   b. This same procedure is used to modify rows.  First delete the rows and then insert the rows.

16. Execute the procedure /shared/ASAssets/KPImetrics/Configuration/**updateAllCacheTables**

   a. Note: this procedure gets executed but will do nothing as caches have been deprecated.

b.  enableDisableFlag – 0=disable, 1=enable, 2=display cache table list

c.  Select 1 to enable all cache tables

## Integrate with DV Metrics

## Configure DV Metrics

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/staging tables. 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 above for details.

> metrics_sessions
> metrics_requests
> metrics_resources_usage

The following 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.

> metrics_sessions → XFER Job → metrics_sessions_hist
> metrics_requests → XFER Job → metrics_requests_hist
> metrics_resources_usage → XFER Job → metrics_resources_usage_hist
> metrics_resources_usage_hist → XFER JOB → METRICS_RESOURCES_USAGE_UD

A diagram of the DBMS Scheduler PLSQL job is provided below.
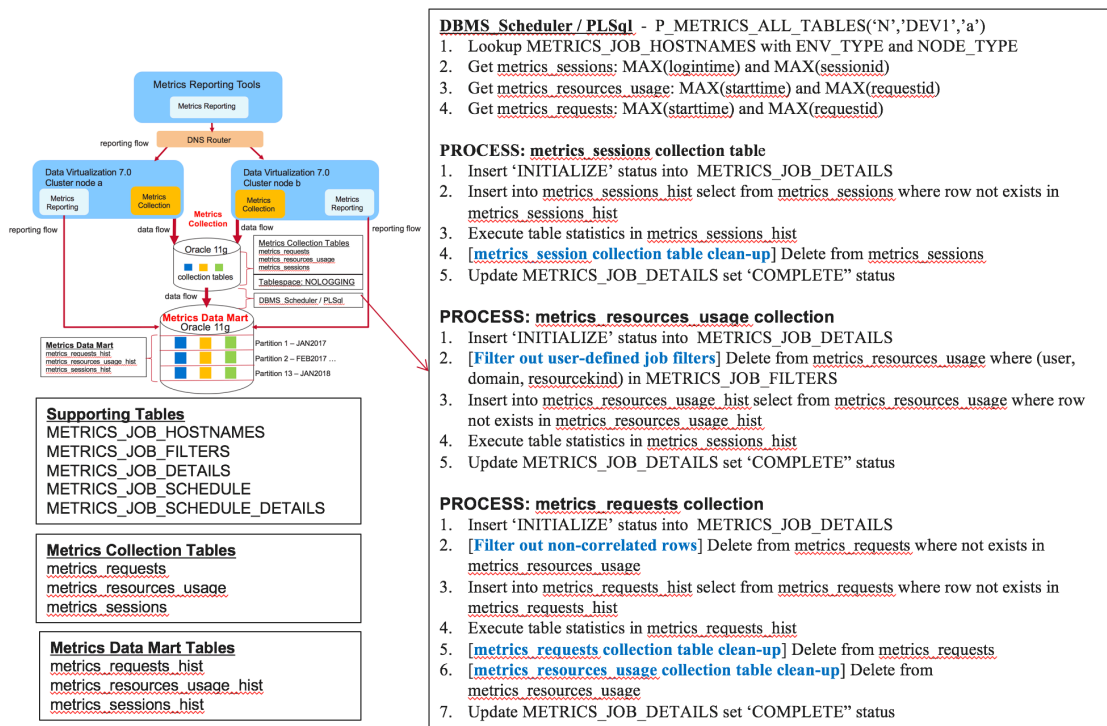
**To enable DV metrics**:

1. Locate and open /policy/metrics
2. Since the tables were created earlier, choose the option to configure the path to the tables without re-creating them.
3. Data Source – browse and set the data source to the database you have configured.
4. Select "Edit Tables"

**Status**

☐ Enable    Buffering Status: ⑦ DISABLED | Truncate Status: ⑦ DISABLED

**Storage**

Data Source:    /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_oracle

Metrics Tables

metrics_sessions
metrics_requests          [ Edit Tables ]
metrics_resources_usage

5. Browse to the schema path
   a. Browse to the Sessions table: metrics_sessions
   b. Browse to the Requests table: metrics_requests
   c. Browse to the Usage table: metrics_resources_usage
   d. DO NOT execute DDL
   e. Click OK to finish

**CIS Create or Bind Tables - Studio 7.0 @ localhost:9400**    ☒

Select Location:  /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_oracle/CIS_KPI    [ Browse... ]

Sessions:    metrics_sessions          [ Browse... ]
Requests:    metrics_requests          [ Browse... ]
Usage:       metrics_resources_usage   [ Browse... ]

[ Execute DDL ]

[ OK ]  [ Cancel ]

6. Enable metrics and save
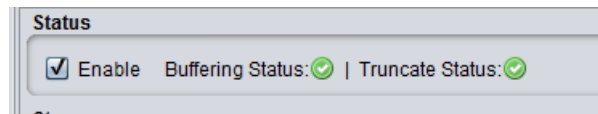
7. If the Buffer Status shows RED then review the following:

   a. Make sure the three collection tables exist in the database and metrics are properly configured to point to them.

   b. Make sure the tables have the correct permissions for writing to them.

   c. If Oracle, make sure the tablespace has the correct permissions for writing to it.

KPImetrics Configuration is COMPLETE!

# 5  KPImetrics Scenarios

**Turn KPI On/Off**

This section describes how to turn KPImetrics on and off by simply turning on/off the triggers.

1. Turn OFF KPImetrics

    a. Execute this procedure
       /shared/ASAssets/KPImetrics/Configuration/updateTriggers

        i. Enable=0

        ii. includeList=null

        iii. excludeList=null

    b. Refresh Studio

    c. Review the triggers status on the Manager tab / Triggers:

| Resource ▲ | Status |
|---|---|
| kpimetricsTrig_00_CheckMetricsActivityDebug | ⑦ Disabled |
| kpimetricsTrig_01_Cache_ALL_RESOURCES | ⑦ Disabled |
| kpimetricsTrig_02_Cache_ALL_USERS | ⑦ Disabled |
| kpimetricsTrig_03_Cache_LDAP_PERSON | ⑦ Disabled |
| kpimetricsTrig_04_Cache_CIS_SYSTEM_RESOURCES | ⑦ Disabled |
| kpimetricsTrig_05_Cache_CPU_MEMORY_CHECKER | ⑦ Disabled |
| kpimetricsTrig_06_Cache_LOG_DISK | ⑦ Disabled |
| kpimetricsTrig_07_Cache_LOG_IO | ⑦ Disabled |
| kpimetricsTrig_08_Cache_LOG_MEMORY | ⑦ Disabled |
| kpimetricsTrig_09_Cache_METRICS_RESOURCES_USAGE_UD | ⑦ Disabled |
| kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST | ⑦ Disabled |
| kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST_REPROCESS | ⑦ Disabled |
| kpimetricsTrig_12_Cache_SYS_CACHES | ⑦ Disabled |
| kpimetricsTrig_13_Cache_SYS_DATASOURCES | ⑦ Disabled |
| kpimetricsTrig_14_CheckCISWorkflowStatusFail | ⑦ Disabled |
| kpimetricsTrig_15_CheckMetricsActivity | ⑦ Disabled |
| kpimetricsTrig_16_PurgeHistoryData | ⑦ Disabled |
| kpimetricsTrig_17_CheckExceedMemoryPercentRequests | ⑦ Disabled |
| kpimetricsTrig_18_CheckLongRunningRequests | ⑦ Disabled |
| kpimetricsTrig_19_AllCustom_AccessByUserOverTime | ⑦ Disabled |
| kpimetricsTrig_20_AllCustom_ActiveResourcesOverPeriodOfTime | ⑦ Disabled |
| kpimetricsTrig_21_AllCustom_ResourceCount_Details | ⑦ Disabled |
| kpimetricsTrig_22_AllCustom_ResourceCount_Total | ⑦ Disabled |
| kpimetricsTrig_30_DBMSScheduler_KPI_oracle | ⑦ Disabled |
| kpimetricsTrig_30_DBMSScheduler_KPI_sqlserver | ⑦ Disabled |
| kpimetricsTrig_31_DBMSSchedulerError_KPI_oracle | ⑦ Disabled |
| kpimetricsTrig_31_DBMSSchedulerError_KPI_sqlserver | ⑦ Disabled |
| kpimetricsTrig_32_DBMSPartitionManager_KPI_oracle | ⑦ Disabled |
| kpimetricsTrig_32_DBMSPartitionManager_KPI_sqlserver | ⑦ Disabled |

2. Turn ON KPImetrics

    a. Execute this procedure
       /shared/ASAssets/KPImetrics/Configuration/updateTriggers
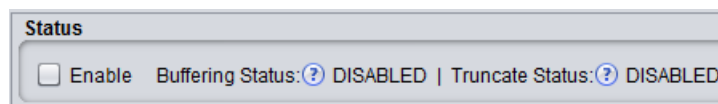
        i. Enable=1

        ii. includeList=null

        iii.   excludeList=null

    b.  Refresh Studio

    c.  Review the triggers status on the Manager tab / Triggers:

| Resource △ | Status |
|---|---|
| kpimetricsTrig_00_CheckMetricsActivityDebug | ⑦ Disabled |
| kpimetricsTrig_01_Cache_ALL_RESOURCES | ✅ Ready |
| kpimetricsTrig_02_Cache_ALL_USERS | ✅ Ready |
| kpimetricsTrig_03_Cache_LDAP_PERSON | ✅ Ready |
| kpimetricsTrig_04_Cache_CIS_SYSTEM_RESOURCES | ✅ Ready |
| kpimetricsTrig_05_Cache_CPU_MEMORY_CHECKER | ✅ Ready |
| kpimetricsTrig_06_Cache_LOG_DISK | ✅ Ready |
| kpimetricsTrig_07_Cache_LOG_IO | ✅ Ready |
| kpimetricsTrig_08_Cache_LOG_MEMORY | ✅ Ready |
| kpimetricsTrig_09_Cache_METRICS_RESOURCES_USAGE_UD | ✅ Ready |
| kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST | ✅ Ready |
| kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST_REPROCESS | ⑦ Disabled |
| kpimetricsTrig_12_Cache_SYS_CACHES | ✅ Ready |
| kpimetricsTrig_13_Cache_SYS_DATASOURCES | ✅ Ready |
| kpimetricsTrig_14_CheckCISWorkflowStatusFail | ✅ Ready |
| kpimetricsTrig_15_CheckMetricsActivity | ✅ Ready |
| kpimetricsTrig_16_PurgeHistoryData | ✅ Ready |
| kpimetricsTrig_17_CheckExceedMemoryPercentRequests | ✅ Ready |
| kpimetricsTrig_18_CheckLongRunningRequests | ✅ Ready |
| kpimetricsTrig_19_AllCustom_AccessByUserOverTime | ✅ Ready |
| kpimetricsTrig_20_AllCustom_ActiveResourcesOverPeriodOfTime | ✅ Ready |
| kpimetricsTrig_21_AllCustom_ResourceCount_Details | ✅ Ready |
| kpimetricsTrig_22_AllCustom_ResourceCount_Total | ✅ Ready |
| kpimetricsTrig_30_DBMSScheduler_KPI_oracle | ✅ Ready |
| kpimetricsTrig_30_DBMSScheduler_KPI_sqlserver | ⑦ Disabled |
| kpimetricsTrig_31_DBMSSchedulerError_KPI_oracle | ✅ Ready |
| kpimetricsTrig_31_DBMSSchedulerError_KPI_sqlserver | ⑦ Disabled |
| kpimetricsTrig_32_DBMSPartitionManager_KPI_oracle | ✅ Ready |
| kpimetricsTrig_32_DBMSPartitionManager_KPI_sqlserver | ⑦ Disabled |

## Turn Data Virtualization (DV) metrics On/Off

This section describes how to turn DV metrics on/off.

1. To Stop DV metrics, you must log in as the "admin" user

    a.  Open /policy/metrics

    b.  Uncheck the "Enable" button

    c.  The metrics will indicate they are DISABLED

**Status**

☐ Enable   Buffering Status:⑦ DISABLED | Truncate Status:⑦ DISABLED

2. To Start DV metrics, you must log in as the "admin" user

    a.  Open /policy/metrics

    b.  Check the "Enable" button

    c.  Both Buffering Status and Truncate Status should be green.
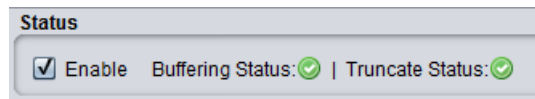
d. If they are not, then there is a problem with the metrics configuration. Check the log entries for errors.

## Modify Triggers

This section describes how to modify the triggers once they are installed.

1. Open/Edit the resource /shared/ASAssets/KPImetrics/Configuration/**defaultTriggersToEnable**

    a. Only modify the ON/OFF settings for each trigger. Leave all other settings alone.

        i. Refer to the following sections for details on each trigger:

            1. Metadata System Triggers and Load Scripts

            2. Physical Oracle Triggers and Scripts

            3. Physical SQL Server Triggers and Scripts

    b. The current triggers defaulted to OFF are as follows:

        i. **kpimetricsTrig_00_CheckMetricsActivityDebug** – Only turn this on if you suspect that DV metrics is not working properly and you want to debug the DV metrics every hour.

        ii. **kpimetricsTrig_12_Cache_METRICS_SQL_REQUEST_REPROCESS** – Only turn this on if you get a code update from the Open Source site and there were changes to the SQL Parser code.

        iii. **kpimetricsTrig_18_CheckExceedMemoryPercentRequests** – You may choose to keep this off in lower-level environments but turn it on in PROD environments.

        iv. **kpimetricsTrig_19_CheckLongRunningRequests** – You may choose to keep this off in lower-level environments but turn it on in PROD environments.

    c. The current triggers defaulted to ON that you may wish to evaluate:

        i. **kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST** – This trigger is defaulted to ON. If you do not wish to perform SQL parsing on the request description SQL statement to parse out the table and column resources used in the SQL then turn this trigger off. There is quite a bit of overhead associated with this trigger.

2. Execute this procedure /shared/ASAssets/KPImetrics/Configuration/updateTriggers

    a. Enable=1

    b. includeList=null

    c. excludeList=null

3. Refresh Studio

4. Review the triggers status on the Manager tab / Triggers to ensure your trigger changes were enforced.

**Perform Oracle Database Maintenance on Collection Tables**

This section outlines how to perform maintenance on the Oracle collection tables in order to regain the tablespace.

1. Stop Data Virtualization metrics

2. Execute the /shared/ASAssets/KPImetrics/Physical/Physical/KPI_oracle/**P_METRICS_ALL_TABLES_exec** to process the existing rows in the collection tables

3. Execute the following commands directly against Oracle

    a. ALTER TABLE metrics_sessions ENABLE ROW MOVEMENT;

    b. ALTER TABLE metrics_sessions SHRINK SPACE CASCADE;

    c. ALTER TABLE metrics_sessions DEALLOCATE UNUSED KEEP 50;

    d. ALTER TABLE metrics_requests ENABLE ROW MOVEMENT;

    e. ALTER TABLE metrics_requests SHRINK SPACE CASCADE;

    f. ALTER TABLE metrics_requests DEALLOCATE UNUSED KEEP 50;

    g. ALTER TABLE metrics_resources_usage ENABLE ROW MOVEMENT;

    h. ALTER TABLE metrics_resources_usage SHRINK SPACE CASCADE;

    i. ALTER TABLE metrics_resources_usage DEALLOCATE UNUSED KEEP 50;

**Configure Third Party Tool Access**

This section outlines how to configure third party tool access for reporting tools such as Cognos, Spotfire, MicroStrategy or others.

1. Download the Data Virtualization (DV) ODBC 7 Drivers and install on the client host machine

2. Configure an ODBC data source

    a. DSN Name: Provide different connections to different DV instances

        i. DV_KPIMETRICS_DEV

        ii. DV_KPIMETRICS_UAT

        iii. DV_KPIMETRICS_PROD

    b. Host – hostname of the DV target instance

    c. Port – port number of the DV target instance (e.g. 9401)

    d. User Name – the user name or service account to use

    e. Password – the password for the user name or service account

    f. Domain – the domain name such as "composite" or "ldap"

    g. Datasource – the datasource will be "ASAssets"

h. Catalog – can leave this blank

i. Local/Code Page – can leave this blank

## Get the Current Row Distribution for the History Tables/Partitions

This section describes how to get the row distribution for the three history tables an there partitions.

1. Oracle – Execute the following procedure with no input:
   /shared/ASAssets/KPImetrics/Physical/Metadata/DDL/Oracle/
   03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_ROW_DISTRIBUTION

2. SQL Server – Execute the following procedure with no input:
   /shared/ASAssets/KPImetrics/Physical/Metadata/DDL/Sqlserver/
   03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_ROW_DISTRIBUTION

# 6    KPImetrics Resources

## Configuration Resources

This section outlines the resources that are used for configuration of KPImetrics.

## KPI Version Overview

Location: /shared/ASAssets/KPImetrics

This section lists the version and "How to Install" procedures.

| Configuration Name | Description |
|---|---|
| getKPIVersion | Returns the version of KPI in the format:  YYYY.Q[1-4]  e.g. 2017.4  If a patch is release within the Quarter then it would be 2017.401 |
| README | Provides a brief description of the published views. |
| RELEASE_NOTES | Provides a description of what changes per release. |

## Configuration Folder Overview

Location: /shared/ASAssets/KPImetrics/Configuration

This section lists all of the configuration procedures that have been defined for the KPImetrics module. These scripts provide functionality for setup/configuration of the KPImetrics module.

| Configuration Name | Description |
|---|---|
| commonValues | Script to store default purge time values and datasource information. |
| getQueryResponseTime | This procedure executes the passed in query and calculates the average time taken to retrieve the first row of data. |
| pMaintenanceUpdateHostnamePort | This procedure is used to convert the values for NODE_HOST/nodehost and NODE_PORT/nodeport from one value to another.  It is "highly" unlikely that this procedure would ever be used but it is provided nonetheless for unforeseen circumstances.  For each view found in /shared/ASAssets/KPImetrics/Physical/Physical/Abstraction perform the updated operation if they contain both NODE_HOST/nodehost and NODE_PORT/nodeport.  This is a dynamic procedure so it will pick up whatever is in that path.<br><br>Note: If you (1) migrate from one host to another host or (2) migrate from one DV instance to another DV instance on the same host, you should not modify the data.  The nodehost and nodeport along with requestid is needed to insure a unique row.  When moving to a new DV instance, the requestid may be repeated. |
| pMetricsEventRegistrationList | This procedure is used to "LIST" to a metrics event registration.   A subscription in the METRICS_EVENT_REGISTRATION table that consists of a unique record for the combination of SUBSCRIBER_EMAIL, GROUP_NAME, ENVIRONMENT_TYPE, EVENT_TYPE and REQUESTER_EMAIL. |
| pMetricsEventRegistrationSubscribe | This procedure is used to "SUBSCRIBE" to a metrics event registration.   A subscription in the METRICS_EVENT_REGISTRATION table consists of a |

| | |
|---|---|
| | unique record for the combination of SUBSCRIBER_EMAIL, GROUP_NAME, ENVIRONMENT_TYPE, EVENT_TYPE and REQUESTER_EMAIL. The column EVENT_TYPE is one of: [LONG_RUNNING\| EXCEEDED_MEMORY\| INACTIVITY\| WORKFLOW_FAILURE\| DBMS_SCHEDULER_ERROR] |
| pMetricsEventRegistrationUnsubscribe | This procedure is used to "UNSUBSCRIBE" to a metrics event registration. A subscription in the METRICS_EVENT_REGISTRATION table that consists of a unique record for the combination of SUBSCRIBER_EMAIL, GROUP_NAME, ENVIRONMENT_TYPE, EVENT_TYPE and REQUESTER_EMAIL. |
| rebindPhysicalAbstraction | The 1$^{st}$ step of this procedure is used to rebind all of the resources (Views) in /shared/ASAssets/KPImetrics/Physical/Physical/Abstraction to the KPI_<database_type> folder as configured in commonValues. All views above the /Abstraction layer will be redirected to use the correct datasource. This is a one-time configuration done during setup. The 2$^{nd}$ step is used to optimize various views and procedures as defined by commonValues.viewOptimizationPathList. <br><br> Some examples of optimization: <br><br> SQL SERVER: <br><br> ex 1. {OPTION IGNORE_TRAILING_SPACES="TRUE", CASE_SENSITIVE="FALSE"} <br><br> ex 2. CAST(DATEDIFF('DAY', requestdate, CURRENT_DATE)/31.00 AS DECIMAL(19,2))  requestdatemonths, CAST(DATEDIFF('DAY', requestdate, CURRENT_DATE) AS DECIMAL(19,0)) requestdatedays, <br><br> ex 3. FROM /shared/ASAssets/KPImetrics/AllCustomReports/ ResourceCount_DetailsRT_sqlserver <br><br> ex 4. FROM /shared/ASAssets/KPImetrics/AllCustomReports/ ResourceCount_TotalRT_sqlserver <br><br> ORACLE: <br><br> ex 1. {OPTION IGNORE_TRAILING_SPACES="FALSE", CASE_SENSITIVE="TRUE"} <br><br> ex 2. CAST(ROUND(MONTHS_BETWEEN(CURRENT_DATE, requestdate),2) AS DECIMAL(19,2)) requestdatemonths, CAST(DAYS_BETWEEN(CURRENT_DATE, requestdate) AS DECIMAL(19,0)) requestdatedays, <br><br> ex 3. FROM /shared/ASAssets/KPImetrics/AllCustomReports/ ResourceCount_DetailsRT_oracle <br><br> ex 4. FROM /shared/ASAssets/KPImetrics/AllCustomReports/ ResourceCount_TotalRT_oracle |
| rebindPhysicalDatabaseType | Rebind the folder /shared/ASAssets/KPImetrics/Physical/Physical/KPI_<database_type> [KPI_oracle, KPI_sqlserver]. Rebind from whatever the current folder is currently pointing to and modify to the new catalog/schema path for tables and procedures. This is required when the catalog and/or the schema name are changed from the original setting. This may be required on the initial setup in the development environment. This may be required if upper environments have a different catalog or schema from the DEV environment although this is not recommended. This procedure is used when the schema and catalog for Oracle |

| | |
|---|---|
| | is not 'CIS_KPI' or SQLServer is not 'CIS_KPI/dbo'.  Configure the following /shared/ASAssets/KPImetrics/Configuration/commonValues.dataSourceName PRIOR to running this procedure. |
| updateAllCacheTables | **DEPRECATED**: All cache tables have been removed from KPImetrics but this procedure will be left here in the event the strategy changes.<br><br>Provides a quick way to disable(0), enable(1), or print status(2) of all the cache tables configured for a given datasource as found in the /KPI_<database_type> folder.  Input is 0,1 or 2 as described previously. |
| updateImpactedResources | Provides a way to iterate through /shared/ASAssets/KPImetrics and attempt to fix any impacted resources due to an anomaly in the DV repository. |
| updateSystemCacheDSandTables | This procedure is used to change the datasource and table that the KPI_<database_type> folder views are using in the cache tab.  This is a one-time configuration executed during setup.<br><br>1. Modify the datasource KPI_<database_type> cache_status and cache_tracking tables to use the correct schema path located here: /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_<database_type><br><br>2. Modify the cached views "Data Source" path and "Table for caching" for the views located here: /shared/ASAssets/KPImetrics/Physical/Physical/KPI_<database_type> Iterate over /shared/ASAssets/KPImetrics/Configuration/defaultTriggersToEnable looking for isCache and cacheTableName.  (a). If cache is configured then configure it but leave the enabled as is.  (b). If cache not configured then configure it and enable it.<br><br>3. Enable the target datasource. |
| updateTriggers | This procedure is used to enable and disable the triggers based on the stored in defaultTriggersToEnable(). |
| defaultTriggersToEnable | This view contains a series of UNION statements for each trigger that exists in the KPImetrics.  It provides information on the following: triggerName, triggerNumber, recommendation, executeImmediate, isCache and cacheTableName.  Example row: 'kpimetricsTrig_01_Cache_ALL_RESOURCES' triggerName,                1 triggerNumber , 'ON' recommendation,   1 executeImmediate, 0 isCache, '' cacheTableName |

## Published Resources

This section outlines the resources that are published under the ASAssets virtual database to expose metrics data. Resources are organized under catalogs and schemas based upon their functionality.

See the section titled KPImetrics Metrics Resources for descriptions of result sets returned by each resource.

## KPImetrics Catalog

### *AllCustomReports Schema*

| Resource (Resource Type) | Description<br><br>**lineage**{pm}=physical metadata table |
|---|---|
| AccessByUserOvertime (TABLE) – cached<br>    AccessByUserOvertimeRT – real time | Cached report of the most active resources by a user over time. Sorted resource count, user and date. The algorithm for this report is shown below. The group by is the key to this report. The key grouping is on the user first and then the resource.<br><br>SELECT<br>"user", "domain", fullname, email, requestdate, resourcepath, resourcename, resourcetype, parentpath, resourcekind, datasourcetype, dataservicename, categoryname, nodehost, nodeport, COUNT(resourceid) countname,<br>CAST(ROUND(MONTHS_BETWEEN(CURRENT_DATE, requestdat),2) AS DECIMAL(19,2)) requestdatemonths,<br>CAST(DAYS_BETWEEN(CURRENT_DATE, requestdate) AS DECIMAL(19,0)) requestdatedays<br>FROM<br>/shared/ASAssets/KPImetrics/Business/Logical/resourceUsage/vResourceUsage<br>GROUP BY "user", "domain", fullname, email, requestdate, resourceid, resourcepath, resourcename, resourcetype, parentpath, resourcekind, datasourcetype, dataservicename, categoryname, nodehost, nodeport<br><br>Derived from the following tables: vResourceCountUsersDate (group by clause) → vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |
| ActiveResourcesOverPeriodOfTime (TABLE) – cached<br>ActiveResourcesOverPeriodOfTimeRT – real time | Cached report of the most active resources over time. Sorted by resource count and date.<br><br>SELECT<br>requestdate, resourcepath, resourcename, resourcetype, parentpath, resourcekind, datasourcetype, dataservicename, categoryname, nodehost, nodeport, COUNT(resourceid) countname,<br>CAST(ROUND(MONTHS_BETWEEN(CURRENT_DATE, requestdate),2) AS DECIMAL(19,2)) requestdatemonths,<br>CAST(DAYS_BETWEEN(CURRENT_DATE, requestdate) AS DECIMAL(19,0)) requestdatedays |

| | |
|---|---|
| | FROM /shared/ASAssets/KPImetrics/Business/Logical/resourceUsage/vResourceUsage GROUP BY ==requestdate==, ==resourceid==, ==resourcepath==, ==resourcename==, ==resourcetype==, ==parentpath==, ==resourcekind==, ==datasourcetype==, ==dataservicename==, ==categoryname==, ==nodehost==, ==nodeport== <br><br> Derived from the following tables: vResourceCountDate (group by clause) → vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |
| ResourceAccessByUsers (TABLE) – cached <br> ResourceAccessByUsersRT – real time | Cached report of the most used resources by a user with no time period. Derived from the following tables: vResourceCountUsers (group by clause) → vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |
| ResourceCount_Details (TABLE) – cached <br> ResourceCount_DetailsRT – real time | Cached detail report of resource count by date. Derived from the following tables: vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |
| ResourceCount_Total (TABLE) – cached <br> ResourceCount_TotalRT – real time | Cached roll-up report of resource count by month. Derived from the following tables: vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |
| SystemCPUandMemoryStatus (TABLE) | Report of system CPU and memory utilization and DV memory over time. Derived from the following tables: vSystemResources → METRICS_CIS_SYSTEM_RESOURCES{pm} |
| vEventRequestSqlColumns (TABLE) | Report of the columns accessed by a SQL request query. Derived from the following tables: [METRICS_SQL_COLUMNS{pm}, METRICS_SQL_RESOURCE{pm}, RequestExpanded --> [metrics_requests_hist{pm}, metrics_sessions_hist{pm}] ] |
| vEventRequestSqlResources (TABLE) | Report of the resources accessed by a SQL request query. Derived from the following tables: [METRICS_SQL_REQUEST{pm}, METRICS_SQL_RESOURCE{pm}, RequestExpanded --> [metrics_requests_hist{pm}, metrics_sessions_hist{pm}] ] |
| vResourceUsage (TABLE) | A raw report of all user defined resources. Derived from the following tables: METRICS_RESOURCES_USAGE_UD{pm} |
| vResourceUsagePublished (TABLE) | A raw report of published user defined resources accessed over time where the resourcekind='user defined' and resourcetype='LINK'. Derived from the following tables: vResourceUsage → METRICS_RESOURCES_USAGE_UD{pm} |

## _cache Schema_

| Resource (Resource Type) | Description |
|---|---|
| vCache (TABLE) | Report of the KPImetrics cache_status table. Displays all records in the table. Derived from METRICS_SYS_CACHES{pm}. |

| | |
|---|---|
| vCacheActive (TABLE) | Report of the KPImetrics cache_status table.  Displays all "**active**" records in the table no matter if the cache is up or down or has a configuration error.  Derived from vCache →METRICS_SYS_CACHES{pm}. |
| vCacheDisabled (TABLE) | Report of the KPImetrics cache_status table.  Displays all "**disabled**" records in the table.  Derived from vCache →METRICS_SYS_CACHES{pm}. |
| vCacheIssues (TABLE) | Report of the KPImetrics cache_status table.  Displays all records with "**issues**" in the table that have an error state such as DOWN, CONFIG ERROR and NOT LOADED.  Derived from vCache →METRICS_SYS_CACHES{pm}. |
| vCacheSchedule (TABLE) | Report of the KPImetrics cache_status table.  Displays all active records in the table with a cache schedule and potential cache schedule dependency.  It is ordered by their next schedule refresh time and dependencies upon other cached resources. Derived from vCache →METRICS_SYS_CACHES{pm}. |

## *configuration Schema*

| Resource (Resource Type) | Description |
|---|---|
| pMetricsEventRegistrationList (PROCEDURE) | Provides the ability to list the subscriptions for the metrics event registration programmatically. |
| pMetricsEventRegistrationSubscribe (PROCEDURE) | Provides the ability to subscribe to a metrics event registration programmatically. |
| pMetricsEventRegistrationUnsubscribe (PROCEDURE) | Provides the ability to unsubscribe to a metrics event registration programmatically. |
| updateTriggers (PROCEDURE) | Provides the ability to turn on/off triggers programmatically. |

## *metrics Schema*

| Resource (Resource Type) | Description |
|---|---|
| **Metrics Collection Tables** | |
| metrics_requests (TABLE) | Native DV out-of-the-box metrics requests collection table. Derived from metrics_requests. |
| metrics_resources_usage (TABLE) | Native DV out-of-the-box metrics resources usage collection table.  Derived from metrics_resources_usage. |
| metrics_sessions (TABLE) | Native DV out-of-the-box metrics sessions collection table.  Derived from metrics_sessions. |
| **Metrics Data Mart History Tables** | |

| | |
|---|---|
| metrics_requests_hist (TABLE) | Historical metrics requests table.  Derived from metrics_requests_hist and metrics_resources_usage_hist.  Expanded with user information and resourcekind and dataservicename. |
| metrics_resources_usage_hist (TABLE) | Historical metrics resources usage table.  Derived from metrics_resources_usage_hist. |
| metrics_sessions_hist (TABLE) | Historical metrics sessions table.  Derived from metrics_sessions_hist and metrics_resources_usage_hist.  Expanded with user information if found. |
| **Metrics Roll-up Counts** | |
| metrics_all_kpimetrics_table_counts (TABLE) | Provides a rollup of all counts by nodehost and nodeport for all metrics tables. |
| metrics_all_min_max_starttime_count (TABLE) | Provides a rollup of the min/max starttime/logintime, min/max requestid/sessionid and the total count of rows for each of the 6 metrics collection and historical tables. |
| **Metrics Requests History Roll-up** | |
| metrics_requests_hist_groupby_date (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by date. |
| metrics_requests_hist_groupby_date_month (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by month. |
| metrics_requests_hist_groupby_date_month_ nodehost_nodeport (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by month, nodehost and nodeport. |
| metrics_requests_hist_groupby_date_user_d omain (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by date, user and domain. |
| metrics_requests_hist_groupby_date_user_d omain_resourcekind (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by date, user, domain and resourcekind [system, user defined]. |
| metrics_requests_hist_groupby_date_user_d omain_resourcekind_dataservicename (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by date, user, domain and resourcekind [system, user defined] and dataservicename. |
| metrics_requests_hist_groupby_nodehost_no deport (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by nodehost and nodeport. |
| metrics_requests_hist_groupby_user_domain _resourcekind_dataservicename (TABLE) | Provides a rollup row count of the metrics_requests_hist table grouped by user, domain, resourcekind [system, user defined] and dataservicename. |
| **Metrics Resources Usage History Roll-Up** | |
| metrics_resources_usage_hist_groupby_date | Provides a rollup row count of the metrics_resources_usage_hist |

| (TABLE) | table grouped by date. |
|---|---|
| metrics_resources_usage_hist_groupby_date _month (TABLE) | Provides a rollup row count of the metrics_resources_usage_hist table grouped by month. |
| metrics_resources_usage_hist_groupby_date _month_nodehost_nodeport (TABLE) | Provides a rollup row count of the metrics_resources_usage_hist table grouped by month, nodehost and nodeport. |
| metrics_resources_usage_hist_groupby_date _user_domain_resourcekind (TABLE) | Provides a rollup row count of the metrics_resources_usage_hist table grouped by date, user, domain and resourcekind [system, user defined]. |
| metrics_resources_usage_hist_groupby_node host_nodeport (TABLE) | Provides a rollup row count of the metrics_resources_usage_hist table grouped by nodehost and nodeport. |
| metrics_resources_usage_hist_groupby_user _domain_resourcekind (TABLE) | Provides a rollup row count of the metrics_resources_usage_hist table grouped by user, domain and resourcekind [system, user defined]. |
| metrics_resources_usage_hist_groupby_user _domain_resourcekind_dataservicename | Provides a rollup row count of the metrics_resources_usage_hist table grouped by user, domain, resourcekind [system, user defined] and dataservicename. |
| **Metrics Sessions History Roll-up** | |
| metrics_sessions_hist_groupby_date (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by date. |
| metrics_sessions_hist_groupby_date_type (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by date and type='end' |
| metrics_sessions_hist_groupby_month (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by month. |
| metrics_sessions_hist_groupby_month_node host_nodeport (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by month, nodehost and nodeport. |
| metrics_sessions_hist_groupby_month_type (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by month and type='end' |
| metrics_sessions_hist_groupby_nodehost_no deport (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by nodehost and nodeport. |
| metrics_sessions_hist_groupby_sessiontype (TABLE) | Provides a rollup row count of the metrics_sessions_hist table grouped by session type [TASK,INTERNAL,JDBC, etc]. |

### _requests Schema_

| Resource (Resource Type) | Description |
|---|---|
| vEventRegLog (TABLE) | Details about the events that have occurred and the emails that have been sent out. An event is only logged if an email is sent. Events are registered in the METRICS_EVENT_REGISRATION table and |

| | |
|---|---|
| | include: LONG_RUNNING, EXCEEDED_MEMORY, INACTIVITY, WORKFLOW_FAILURE, DBMS_SCHEDULER_ERROR, and PURGE_HISTORY. |
| vEventRegLogLineage (TABLE) | Details about the data source lineage for a SQL description (request) that results from an event registered by METRICS_EVENT_REGISTRATION.  Events that log data source lineage include: LONG_RUNNING and EXCEEDED_MEMORY because they have an associated SQL description that gets parsed in order to determine the lineage. |
| vEventRequestSqlColumns (TABLE) | Details about the column projection list from the metrics_request description (SQL).  These columns were parsed.  Derived from the following tables: METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE, METRICS_SQL_COLUMNS, metrics_requests_hist and metrics_sessions_hist |
| vEventRequestSqlResources (TABLE) | Details about the resources used from the metrics_request description (SQL).  These resources were parsed.  Derived from the following tables: METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE, metrics_requests_hist and metrics_sessions_hist |
| vEventRequestSqlResourcesAllErrors (TABLE) | Details about errors produced during parsing of the metrics_request description (SQL).  This can be used to improve the SQL Parser code implemented by KPImetrics.  These resources were parsed.  Derived from the following tables: METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE, metrics_requests_hist and metrics_sessions_hist |
| vEventRequestSqlResourcesCount (TABLE) | Returns a count of the number of unique requestid rows parsed for each metrics_request description (SQL).  These resources were parsed.  Derived from the following tables: METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE, metrics_requests_hist and metrics_sessions_hist |
| vExceededMemoryPercentRequests (TABLE) | Details on exceede memory queries that are occurring at the time this resource is executed.  Derived from the system catalog table SYS_REQUESTS. |
| vGetSystemInformation (TABLE) | Returns nodehost and nodeport for a DV server. Derived from /lib/util/GetProperty('SERVER_HOSTNAME') and |

| | |
|---|---|
| | /lib/util/GetProperty('SERVER_JDBC_PORT') |
| vLongRunningRequests (TABLE) | Details on long running queries that are occurring at the time this resource is executed. Derived from the system catalog table SYS_REQUESTS. |
| vMetricsSqlColumns (TABLE) | Details of the parsed SQL columns for a user defined request. Derived from METRICS_SQL_COLUMNS with no other join. |
| vMetricsSqlRequest (TABLE) | Details of the parsed SQL for a user defined request. Derived from METRICS_SQL_REQUEST with no other join. |
| vMetricsSqlRequestUniqueSqlTemplates (TABLE) | A unique grouping of parsed SQL templates derived from parsing the request "description". Warning. Since the description is stored as a CLOB field and push down on CLOB is not supported, this may cause high memory usage in CIS. |
| vMetricsSqlRequestUniqueSqlTemplatesByUser (TABLE) | A unique grouping of parsed SQL templates sorted by user and derived from parsing the request "description". Warning. Since the description is stored as a CLOB field and push down on CLOB is not supported, this may cause high memory usage in CIS. |
| vMetricsSqlRequestUniqueSqlTemplatesByUserByDate (TABLE) | A unique grouping of parsed SQL templates sorted by user and request date and derived from parsing the request "description". Warning. Since the description is stored as a CLOB field and push down on CLOB is not supported, this may cause high memory usage in CIS. |
| vMetricsSqlResource (TABLE) | Details of the parsed SQL resources for a user defined request. Derived from METRICS_SQL_RESOURCE with no other joins. |
| vPublishedResourcePerRequest (TABLE) | Details on published requests correlated with user information. Derived from METRICS_RESOURCES_USAGE_UD, metrics_sessions_hist, metrics_requests_hist. |
| vRequestExpandedAll (TABLE) | Details of a request expanded with user information. All records are displayed. Derived from metrics_requests_hist and metrics_sessions_hist. |
| vRequestExpandedUD (TABLE) | Details of a request expanded with user information. Only records of resourcekind='user defined' are displayed. Derived from metrics_requests_hist and metrics_sessions_hist. |
| vRequestsCountsByUser (TABLE) | A count of requests by user and date along with the following per request: avg rows, avg bytes from |

| | |
|---|---|
| | client, avg bytes to client, min total duration, max total duration, min server duration and max server duration. |
| vSessions (TABLE) | Details for user sessions. Equivalent to metrics_sessions_hist. |
| vSessionsUserRequests (TABLE) | Details on requests generated by each user session |

### resource Schema

| Resource (Resource Type) | Description |
|---|---|
| vAllResourcesHist (TABLE) | Report of all resources historically in the DV repository. Derived from METRICS_ALL_RESOURCES and is loaded only with new resources based on a trigger. |
| vAllResourcesMax (TABLE) | Report of the latest resources in the DV repository. Derived from METRICS_ALL_RESOURCES. |
| vResourceCount (TABLE) | Report of the count of resources where resourcekind='user defined'. Derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceCountDate (TABLE) | Report of the count of resources by date where resourcekind='user defined'. Derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceCountUsers (TABLE) | Report of the count of resources by user where resourcekind='user defined'. Derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceCountUsersDate (TABLE) | Report of the count of resources by user and date where resourcekind='user defined'. Derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceDistinctPublishedDatabases (TABLE) | This table returns a list of all resources published under a database on the DV server derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceDistinctPublishedResources (TABLE) | This table returns a list of all distinct resources published under a database on the DV server derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceDistinctPublishedWebServices (TABLE) | This table returns a list of all resources published as a web service operation on the DV server derived from METRICS_RESOURCES_USAGE_UD. |
| vResourceDistinctResources (TABLE) | This table returns a list of all distinct resources on the DV server derived from METRICS_RESOURCES_USAGE_UD. |
| vResourcesPublishedNotUsed (TABLE) | Report of all published resources present on the DV server that have not been used derived from |

| | METRICS_ALL_RESOURCES, METRICS_RESOURCES_USAGE_UD. |
|---|---|
| vResourceUsageAll (TABLE) | Report of all resources historically where resourcekind='user defined' and 'system'. Derived from METRICS_ALL_RESOURCES, metrics_resources_usage_hist. |
| vResourceUsageUD (TABLE) | Report of all resources where resourcekind='user defined'. Derived from METRICS_RESOURCES_USAGE_UD. |

### resourceDataCount Schema

| Resource (Resource Type) | Description |
|---|---|
| getResourceDataCount (PROCEDURE) | This procedure returns a list of the top N most frequently accessed resources for the specified data range. Each row includes a count of the number of rows of data each resource contains |

### resourceMetadata Schema

| Resource (Resource Type) | Description |
|---|---|
| vResourceListAllPublishedResources (TABLE) | This table returns a list of all published resources present on the DV server metadata catalog derived from the cached METRICS_ALL_RESOURCES. |

### systemUsage Schema

| Resource (Resource Type) | Description |
|---|---|
| vCpuMemUtilization (TABLE) | Details on system CPU and memory utilization. Derived from METRICS_CPU_MEMORY_CHECKER. |
| vDatasourceConnectionChanges (TABLE) | Details on data sources connection changes. Derived from METRICS_SYS_DATASOURCES which was cached from /services/databases/system/SYS_DATASOURCES. |
| vDatasourceStatusChanges (TABLE) | Details on data source status changes. Derived from METRICS_SYS_DATASOURCES which was cached from /services/databases/system/SYS_DATASOURCES. |
| vDatasourceUsage (TABLE) | Details on data sources usage. Derived from METRICS_SYS_DATASOURCES which was cached from /services/databases/system/SYS_DATASOURCES. |
| vDatasourceUsageCurrent (TABLE) | Details on current data sources usage. Derived from METRICS_SYS_DATASOURCES which was cached from |

| | |
|---|---|
| | /services/databases/system/SYS_DATASOURCES |
| vLogDisk (TABLE) | This table returns DV server disk events derived from the cached METRICS_LOG_DISK which was cached from /services/databases/system/LOG_DISK. The information includes configured disk size/used, temporary disk size/used and log disk size/used. |
| vLogIO (TABLE) | This table returns DV server disk events derived from the cached METRICS_LOG_IO which was cached from /services/databases/system/LOG_IO. The information includes bytes from clients, bytes to clients, bytes from data sources and bytes to data sources. |
| vLogMemory (TABLE) | This table returns DV server disk events derived from the cached METRICS_LOG_MEMORY which was cached from /services/databases/system/LOG_MEMORY. The information memory bytes, memory max, managed bytes and managed max. |
| vSystemResources (TABLE) | Details on system resource usage. Derived from METRICS_CIS_SYSTEM_RESOURCES. |

## *users Schema*

| Resource (Resource Type) | Description |
|---|---|
| vAllUsersHist (TABLE) | Details on all users over time derived from METRICS_ALL_USERS. |
| vAllUsersMax (TABLE) | Details on the latest user derived from METRICS_ALL_USERS. |
| vLdapPerson (TABLE) | Details regarding an LDAP user derived from METRICS_LDAP_PERSON. |

## *workflow Schema*

| Resource (Resource Type) | Description |
|---|---|
| vCISWorkflow (TABLE) | Report to view the status of the KPImetrics triggers derived from METRICS_CIS_WORKFLOW. |
| vCISWorkflowStatus (TABLE) | Report to view the status of the KPImetrics triggers sort by workflowstatus ASC and derived from METRICS_CIS_WORKFLOW. Failure message come first. Workflowstatus=F, I, S. F=Fail, I=In process, S=Success. |
| vEventRegistration (TABLE) | Report to view the list of email event registration subscriptions that are currently configured. |
| vJobDetails (TABLE) | Report to view the status of the METRICS_JOB_DETAILS DBMS Scheduler table. Provides a sorted status by most |

| | recent rows first.  The information informs the user of the status of the data transfer from the collection tables to the history tables for each of the 3 metrics tables. |
|---|---|
| vJobEnvironments (TABLE) | Report to view the list of valid environments. |
| vJobFilters (TABLE) | Report to view the list of job filters that are currently configured. |

## Data Sources

This section outlines the data sources created, populated and used by KPImetrics project.

### Metadata Data Source for LDAP

Location: /shared/ASAssets/KPImetrics/Physical/Metadata/LDAP

The data source LDAP is an LDAP data source that connects to a client's corporate LDAP directory to lookup user information and their relation to client hierarchy. The data source's URL property should be modified to allow the data source to successfully connect to and query the target LDAP directory server. The LDAP structure organizationalPerson must be introspected under this data source for LDAP integration to function successfully.

*It is strongly recommended that this data source should not use the same LDAP account as the DV server uses to authenticate LDAP users. This may result in the LDAP account being locked if the data source's credentials are not updated when the account's password is changed.*

### Metadata Data Source for CPUAndMemChecker

Location: /shared/ASAssets/KPImetrics/Physical/Metadata/CPUAndMemChecker

CPUAndMemChecker custom java procedure is used to capture system level CPU and Memory usage at the operating system level. On a linux server installation, CPUAndMemChecker invokes two shell scripts (KPImetricsTopCommandGrepCpu_linux7.sh and KPImetricsFreeMemCommand_linux7.sh) to execute 'top' and 'free' commands to returns CPU percentage, used memory and available memory. Windows installations use a couple of powershell scripts (KPImetricsCpuUtilization.ps1 and KPImetricsMemUtilization.ps1) to perform the same capability.

The CPUAndMemChecker procedure is invoked by Cache_CIS_SYSTEM_RESOURCES script and inserts the results in METRICS_CIS_SYSTEM_RESOURCES table of the KPImetrics data source.

The CPUAndMemChecker procedure exposes one procedure that has following parameters:

| Parameter Name | Direction | Description |
|---|---|---|
| debug | IN | Y=debug values written to cs_server.log.  N=no debug. |
| cpuScriptNameOrCommand | IN | Path to script to return CPU utilization.<br><br>Windows example:  powershell.exe -file C:\CIS7\bin\KPImetricsCpuUtilization.ps1<br><br>UNIX example: /CIS7/bin/KPImetricsTopCommandGrepCpu_linux7.sh |
| memScriptNameOrCommand | IN | Path to script to return memory utilization.<br><br>Windows example: powershell.exe -file C:\CIS7\bin\KPImetricsMemUtilization.ps1<br><br>UNIX example: |

| | | /CIS7/bin/KPImetricsFreeMemCommand_linux7.sh |
|---|---|---|
| cpuUsedPercent | OUT | Average CPU utilization percentage reported by the server's operating system |
| memoryUsedMb | OUT | Used memory in Megabytes reported by the server's operating system |
| memoryAvailMb | OUT | Available memory in Megabytes reported by the server's operating system |

**Metadata Data Source for KPI_<database_type>**

Location: /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_<database_type>

The KPImetrics module provides data source for all currently supported storage database platforms under /shared/ASAssets/KPImetrics/Physical/Metadata.

Currently the KPImetrics module includes the following KPImetrics data sources

- /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_oracle
- /shared/ASAssets/KPImetrics/Physical/Metadata/KPI_sqlserver

During deployment, the KPImetrics module must be configured to use the data source appropriate for the target KPImetrics database platform. The following instructions refer to this data source at the KPImetrics data source for simplicity

The KPImetrics data source is used to capture

- Historical server metrics captured using incremental caching
- Some pre-processed data for final reporting

The following tables have been created in CIS_KPI schema to capture the required data.

| Table Name | Description |
|---|---|
| P_METRICS_ALL_TABLES | This procedure is used by DV to execute the native database data transfer procedure which formats and moves data from the metrics collection tables to the metrics history tables on a scheduled basis.<br>**Oracle Lineage**:<br>TRIGGER:<br>/KPImetrics/Physical/Physical/KPI_oracle/ kpimetricsTrig_30_DBMSScheduler_KPI_oracle →<br>PROCEDURES:<br>/KPImetrics/Physical/Physical/KPI_oracle/P_METRICS_ALL_TABLES_exec →<br>/KPImetrics/Physical/Physical/KPI_oracle/P_METRICS_ALL_TABLES →<br>/KPImetrics/Physical/Metadata/KPI_oracle/<schema>/P_METRICS_ALL_TABLES<br>**SQL Server Lineage**:<br>TRIGGER:<br>/KPImetrics/Physical/Physical/KPI_sqlserver/ kpimetricsTrig_30_DBMSScheduler_KPI_sqlserver →<br>PROCEDURES:<br>/KPImetrics/Physical/Physical/KPI_sqlserver/P_METRICS_ALL_TABLES_exec →<br>/KPImetrics/Physical/Physical/KPI_sqlserver/P_METRICS_ALL_TABLES →<br>/KPImetrics/Physical/Metadata/KPI_sqlserver/<catalog>/<schema>/ P_METRICS_ALL_TABLES |
| P_REPAIR_METRICS_REQUESTS_HIST | This procedure is similar to P_METRICS_ALL_TABLES except that it only repairs the metrics_request_hist table.  It adds in the extra LDAP information, resourcekind and dataservicename. Currently only supported on Oracle. |
| cache_status | This table is used by DV to manage incremental caches used by the KPImetrics module. Developers should not modify the table or its contents unless explicitly instructed by DV |

| | |
|---|---|
| | support or Advanced Services. |
| cache_tracking | This table is used by DV to manage incremental caches used by the KPImetrics module. Developers should not modify the table or its contents unless explicitly instructed by DV support or Advanced Services. |
| METRICS_ACR_ABUOT | The cache table acronym "METRICS_ACR_ABUOT" must be short and thus it stands for "All Custom Reports Access By User Over Time". It is cached once a day from the real-time query "ACR_AccessByUserOvertime". It is access from AllCustomReports.AccessByUserOvertime. <br><br> TRIGGER: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_19_AllCustom_AccessByUserOverTime → <br> PROCEDURE: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ Cache_AllCustom_AccessByUserOverTime → <br> READ: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ACR_AccessByUserOverTime → <br> INSERT: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ACR_ABUOT |
| METRICS_ACR_AROPOT | The cache table acronym "METRICS_ACR_AROPOT" must be short and thus it stands for "All Custom Reports Active Resources Over Period Of Time". It is cached once a day from the real-time query "ACR_ActiveResourcesOverPeriodOfTime". It is access from AllCustomReports.ActiveResourcesOverPeriodOfTime. <br><br> TRIGGER: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_20_AllCustom_ActiveResourcesOverPeriodOfTime → <br> PROCEDURE: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ Cache_AllCustom_ActiveResourcesOverPeriodOfTime → <br> READ: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ ACR_ActiveResourcesOverPeriodOfTime → <br> INSERT: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ACR_AROPOT |
| METRICS_ACR_RCD | The cache table acronym "METRICS_ACR_RCD" must be short and thus it stands for "All Custom Reports Resource Count Details". It is cached once a day from the real-time query "ACR_ResourceCount_Details". It is access from AllCustomReports.ResourceCount_Details. <br><br> TRIGGER: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_21_AllCustom_ResourceCount_Details → <br> PROCEDURE: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ Cache_AllCustom_ResourceCount_Details → <br> READ: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ACR_ResourceCountDetails → <br> INSERT: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ACR_RCD |
| METRICS_ACR_RCT | The cache table acronym "METRICS_ACR_RCT" must be short and thus it stands for "All Custom Reports Resource Count Total". It is cached once a day from the real-time query "ACR_ResourceCount_Total". It is access from AllCustomReports.ResourceCount_Total. <br> TRIGGER: |

| | |
|---|---|
| | /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_22_AllCustom_ResourceCount_Total → |
| | <u>PROCEDURE</u>: |
| | /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ Cache_AllCustom_ResourceCount_Total→ |
| | <u>READ</u>: |
| | /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ACR_ResourceCount_Total → |
| | <u>INSERT</u>: |
| | /KPImetrics/Physical/Physical/Abstraction/METRICS_ACR_RCT |
| METRICS_ALL_RESOURCES | This table stores a cached representation of the DV system ALL_RESOURCES table because it is more efficient to query.  It is processed using the system interface lineage: <br> <u>TRIGGER</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_01_Cache_ALL_RESOURCES → <br> <u>PROCEDURE</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_ALL_RESOURCES → <br> <u>READ</u>: <br> /KPImetrics/Physical/Metadata/System/ALL_RESOURCES → <br> <u>INSERT</u>: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ALL_RESOURCES_STG → <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ALL_RESOURCES |
| METRICS_ALL_RESOURCES_STG | This table is used to stage the data being inserted into METRICS_ALL_RESOURCES. |
| METRICS_ALL_USERS | This table stores user information for all user accounts that have executed queries against the DV instance. This is an incremental cache target table.  It is processed using the system interface lineage: <br> <u>TRIGGER</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_02_Cache_ALL_USERS → <br> <u>PROCEDURE</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_ALL_USERS → <br> <u>READ</u>: <br> /KPImetrics/Physical/Metadata/System/ALL_USERS → <br> <u>INSERT</u>: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_ALL_USERS |
| METRICS_CIS_SYSTEM_RESOURCES | This tables stores hourly snapshots of memory, disk and I/O usage captured from SYS_MEMORY, SYS_STORAGE and SYS_IO tables.  It is processed using the system interface lineage: <br> <u>TRIGGER</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_04_Cache_CIS_SYSTEM_RESOURCES→ <br> <u>PROCEDURE</u>: <br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ Cache_CIS_SYSTEM_RESOURCES→ <br> <u>READ</u>: <br> /KPImetrics/Physical/Physical/Abstraction/METRICS_CIS_SYSTEM_RESOURCES + /KPImetrics/Physical/Metadata/System/Helpers/p15MinutesIncrements <br> <u>INSERT</u>: |

| | /KPImetrics/Physical/Physical/Abstraction/METRICS_CIS_SYSTEM_RESOURCES |
|---|---|
| METRICS_CIS_WORKFLOW | The CIS_WORKFLOW table manages workflows for processing data.<br><br>PROCEDURE:<br>/KPImetrics/Physical/Metadata/System/Helpers/pStartWorkflow<br>/KPImetrics/Physical/Metadata/System/Helpers/pEndWorkflow<br><br>INSERT:<br>/KPImetrics/Physical/Physical/Abstraction/METRICS_CIS_WORKFLOW |
| METRICS_CPU_MEMORY_CHECKER | This table stores the results of executing the CPUAndMemChecker procedure over time. This is an incremental cache target table.  It is processed using the system interface lineage:<br><br>TRIGGER:<br>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>    kpimetricsTrig_05_Cache_CPU_MEMORY_CHECKER→<br><br>PROCEDURE:<br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/<br>    Cache_CPU_MEMORY_CHECKER→<br><br>READ:<br>/KPImetrics/Physical/Metadata/System/CPU_MEMORY_CHECKER →<br>/KPImetrics/Physical/Metadata/CPUAndMemChecker/CpuAndMemCheckerCjp →<br><br>CPU Utilization<br>KPImetricsCpuUtilization.ps1<br>KPImetricsTopCommandGrepCpu_linux6.sh<br>KPImetricsTopCommandGrepCpu_linux7.sh<br><br>Memory Utilization<br>KPImetricsMemUtilization.ps1<br>KPImetricsFreeMemCommand_linux6.sh<br>KPImetricsFreeMemCommand_linux7.sh<br><br>INSERT:<br>/KPImetrics/Physical/Physical/Abstraction/METRICS_CPU_MEMORY_CHECKER |
| METRICS_EVENT_REG_LOG | This table stores the details about the events that have occurred and the emails that have been sent out.  An event is only logged if an email is sent.  Events are registered in the METRICS_EVENT_REGISRATION table and include: LONG_RUNNING, EXCEEDED_MEMORY, INACTIVITY and PURGE_HISTORY.  The event time, the user, actual email along with the SQL description when applicable is stored.<br><br>TRIGGER:<br>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>1. kpimetricsTrig_15_CheckMetricsActivity → Event=INACTIVITY<br>2. kpimetricsTrig_16_PurgeHistoryData → Event=PURGE_HISTORY<br>3. kpimetricsTrig_17_CheckExceedMemoryPercentRequests →<br>    Event=EXCEEDED_MEMORY<br>4. kpimetricsTrig_18_CheckLongRunningRequests →Event=LONG_RUNNING<br><br>PROCEDURE:<br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/<br>1. pCheckMetricsActivity → ../Helpers/pUpdateEventRegLog<br>2. pPurgeData → ../Helpers/pUpdateEventRegLog<br>3. pCheckExceededMemoryPercentRequests → ../Helpers/pUpdateEventRegLog<br>4. pCheckLongRunningRequests → ../Helpers/pUpdateEventRegLog<br><br>READ: /shared/ASAssets/KPImetrics/Physical/Metadata/System/ |

| | |
|---|---|
| | 1. METRICS_JOB_DETAILS<br><br>2. N/A<br><br>3. /KPImetrics/Business/Business/requests/vExceededMemoryPercentRequests<br><br>4. /KPImetrics/Business/Business/requests/vLongRunningRequests<br><br>INSERT:<br><br>/KPImetrics/Physical/Physical/Abstraction/METRICS_EVENT_REG_LOG and<br>    METRICS_EVENT_REG_LOG_LINEAGE |
| METRICS_EVENT_REG_LOG_LINEAGE | This table is a child table to METRICS_EVENT_REG_LOG and is used to store the data source lineage for the SQL request that was logged as a result of either LONG_RUNNING or EXCEEDED_MEMORY.  The other events do not produce a SQL description so are not applicable.  A SQL description is parsed using the same procedures that produce the METRICS_SQL_REQUEST data and store one or more rows associated with the requests data source and connection information.  The reporting on this can look for events and quickly determine which data source is being accessed which can assist the viewer on how to take action with the data owners if necessary. |
| METRICS_EVENT_REGISTRATION | This table is used to register a metrics event registration.   A subscription in the METRICS_EVENT_REGISTRATION table consists of a unique record for the combination of SUBSCRIBER_EMAIL, GROUP_NAME, ENVIRONMENT_TYPE, EVENT_TYPE and REQUESTER_EMAIL.  The EVENT_TYPE can be one of [LONG_RUNNING| EXCEEDED_MEMORY| INACTIVITY| WORKFLOW_FAILURE| DBMS_SCHEDULER_ERROR]. |
| METRICS_JOB_DETAILS | This table is used to hold the data transfer job details when rows are moved from the metrics collection tables to the history tables.  The native database procedure "P_METRICS_ALL_TABLES" performs the following data transfer capabilities:<br><br>metrics_sessions → metrics_sessions_hist<br><br>metrics_resources_usage → metrics_resources_usage_hist<br><br>metrics_requests → metrics_requests_hist |
| METRICS_JOB_ENVIRONMENTS | This table provides a list of valid environments.  In essence, the ENV_TYPE is like the short nickname for a host.  For example, DEV1 is the short-name the development server.   This is used by various email notification procedures. |
| METRICS_JOB_FILTERS | This table is used to hold the job filters used by "P_METRICS_ALL_TABLES" whereby the metrics_resource_usage collection table filters out rows based on user, domain and resourcekind.  This capability allows rows to be filtered out before they get to the history table thus reducing the overall burden.  Without this feature, the database would be overwhelmed by millions of unnecessary rows as absolutely everything in DV is reported. |
| METRICS_LDAP_PERSON | This LDAP_PERSON table is used to pre-cache LDAP user information.  It is processed using the system interface lineage:<br><br>TRIGGER:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>    kpimetricsTrig_03_Cache_LDAP_PERSON→<br><br>PROCEDURE:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_LDAP_PERSON→<br><br>READ:<br><br>/KPImetrics/Physical/Metadata/System/LDAP_PERSON<br><br>INSERT:<br><br>synchronous cache refresh executed on one of applicable tables:<br><br>/KPImetrics/Physical/Physical/KPI_oracle/METRICS_LDAP_PERSON<br><br>/KPImetrics/Physical/Physical/KPI_sqlserver/METRICS_LDAP_PERSON |
| METRICS_LOG_DISK | This table stores logs of available disk space incrementally cached from the DV system table |

| | |
|---|---|
| | LOG_DISK. It is processed using the system interface lineage: TRIGGER: /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/    kpimetricsTrig_06_Cache_LOG_DISK→ PROCEDURE: /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_LOG_DISK→ READ: /KPImetrics/Physical/Metadata/System/LOG_DISK INSERT: /KPImetrics/Physical/Physical/Abstraction/METRICS_LOG_DISK |
| METRICS_LOG_IO | This table stores IO logs incrementally cached from the DV system table LOG_IO. It is processed using the system interface lineage: TRIGGER: /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/    kpimetricsTrig_07_Cache_LOG_IO→ PROCEDURE: /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_LOG_IO→ READ: /KPImetrics/Physical/Metadata/System/LOG_IO INSERT: /KPImetrics/Physical/Physical/Abstraction/METRICS_LOG_IO |
| METRICS_LOG_MEMORY | This table stores jvm memory logs incrementally cached from the DV system table LOG_MEMORY. It is processed using the system interface lineage: TRIGGER: /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/    kpimetricsTrig_08_Cache_LOG_MEMORY → PROCEDURE: /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_LOG_MEMORY → READ: /KPImetrics/Physical/Metadata/System/LOG_MEMORY INSERT: /KPImetrics/Physical/Physical/Abstraction/METRICS_LOG_MEMORY |
| metrics_requests | This is the DV out-of-the-box requests table. It cannot contain any indexes. Its purpose is simply to be a collector of DV metrics requests. |
| metrics_requests_hist | This is the KPImetrics historical requests table. It may be partitioned and contain indexes for better query performance. It is updated periodically from the metrics_requests table. If partitioned by month, it allows a more efficient way to purge data by dropping a partition rather than deleting data. |
| metrics_requests_stg | This is a KPImetrics historical staging table that is used during the data transfer procedure for doing a mass update with Oracle. |
| metrics_resources_usage | This is the DV out-of-the-box resources usage table. It cannot contain any indexes. Its purpose is simply to be a collector of DV metrics resources usage. |
| metrics_resources_usage_hist | This is the KPImetrics historical resources usage table. It may be partitioned and contain indexes for better query performance. It is updated periodically from the metrics_resources_usage table. If partitioned by month, it allows a more efficient way to purge data by dropping a partition rather than deleting data. |
| METRICS_RESOURCES_USAGE_UD | This is a pre-processed table of metrics_resources_usage_hist that contains only |

| | |
|---|---|
| | "USER_DEFINED" queries along with user information, data service name and category name used as filters.  The view "vResourceUsage" sits on top of this view.  It is processed using the system interface lineage:<br><br>TRIGGER:<br><br>KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>    kpimetricsTrig_09_Cache_METRICS_RESOURCES_USAGE_UD →<br><br>PROCEDURE:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/<br>    Cache_METRICS_RESOURCES_USAGE_UD →<br><br>READ:<br><br>/KPImetrics/Physical/Physical/Abstraction/metrics_resources_usage_hist<br><br>INSERT:<br><br>/KPImetrics/Physical/Physical/Abstraction/METRICS_RESOURCES_USAGE_UD |
| metrics_sessions | This is the DV out-of-the-box sessions table.  It cannot contain any indexes.  Its purpose is simply to be a collector of DV metrics sessions. |
| metrics_sessions_hist | This is the KPImetrics historical sessions table.  It may be partitioned and contain indexes for better query performance.  It is updated periodically from the metrics_sessions table.  If partitioned by month, it allows a more efficient way to purge data by dropping a partition rather than deleting data. |
| METRICS_SQL_COLUMNS | This table holds the parsed SQL columns for the query statement.  It is processed using the system interface lineage:<br><br>TRIGGER:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>    kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST and<br>    kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST_REPROCESS→<br><br>PROCEDURE:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/<br><br>Cache_METRICS_SQL_REQUEST_EXEC →<br>    Cache_METRICS_SQL_REQUEST_GENSQL and<br><br>Cache_METRICS_SQL_REQUEST_EXEC_REPROCESS→<br>    Cache_METRICS_SQL_REQUEST_GENSQL_REPROCESS<br><br>READ:<br><br>/KPImetrics/Physical/Formatting/metrics_requests_hist +<br>    /KPImetrics/Physical/Formatting/METRICS_SQL_REQUEST<br><br>INSERT:<br><br>/KPImetrics/Physical/Physical/Abstraction/METRICS_SQL_COLUMNS |
| METRICS_SQL_REQUEST | This table holds the parsed SQL status for the query statement.  It is processed using the system interface lineage:<br><br>TRIGGER:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/<br>    kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST and<br>    kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST_REPROCESS→<br><br>PROCEDURE:<br><br>/KPImetrics/Physical/Metadata/System/ClusterSafeCache/<br><br>Cache_METRICS_SQL_REQUEST_EXEC →<br>    Cache_METRICS_SQL_REQUEST_GENSQL and<br><br>Cache_METRICS_SQL_REQUEST_EXEC_REPROCESS→<br>    Cache_METRICS_SQL_REQUEST_GENSQL_REPROCESS<br><br>READ: |

| | /KPImetrics/Physical/Formatting/metrics_requests_hist + /KPImetrics/Physical/Formatting/METRICS_SQL_REQUEST <br><br> INSERT: <br><br> /KPImetrics/Physical/Physical/Abstraction/METRICS_SQL_REQUEST |
|---|---|
| METRICS_SQL_RESOURCE | This table holds the parsed SQL resource for the query statement.  It is processed using the system interface lineage: <br><br> TRIGGER: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST and kpimetricsTrig_12_Cache_METRICS_SQL_REQUEST_REPROCESS→ <br><br> PROCEDURE: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/ <br><br> Cache_METRICS_SQL_REQUEST_EXEC → Cache_METRICS_SQL_REQUEST_GENSQL and <br><br> Cache_METRICS_SQL_REQUEST_EXEC_REPROCESS→ Cache_METRICS_SQL_REQUEST_GENSQL_REPROCESS <br><br> READ: <br><br> /KPImetrics/Physical/Formatting/metrics_requests_hist + /KPImetrics/Physical/Formatting/METRICS_SQL_REQUEST <br><br> INSERT: <br><br> /KPImetrics/Physical/Physical/Abstraction/METRICS_SQL_RESOURCE |
| METRICS_SYS_CACHES | This table stores a list of all cached resources and their current statuses incrementally cached from the DV system table SYS_CACHES.  It is processed using the system interface lineage: <br><br> TRIGGER: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_12_Cache_SYS_CACHES → <br><br> PROCEDURE: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_SYS_CACHES → <br><br> READ: <br><br> /KPImetrics/Physical/Metadata/System/SYS_CACHES <br><br> INSERT: <br><br> /KPImetrics/Physical/Physical/Abstraction/METRICS_SYS_CACHES |
| METRICS_SYS_DATASOURCES | This table stores a list of all data sources and their current statuses incrementally cached from the DV system table SYS_DATASOURCES.  It is processed using the system interface lineage: <br><br> TRIGGER: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_13_Cache_SYS_DATASOURCES → <br><br> PROCEDURE: <br><br> /KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_SYS_DATASOURCES → <br><br> READ: <br><br> /KPImetrics/Physical/Metadata/System/SYS_DATASOURCES <br><br> INSERT: <br><br> /KPImetrics/Physical/Physical/Abstraction/METRICS_SYS_DATASOURCES |

## Metadata System Triggers and Load Scripts

Location: /shared/ASAssets/KPImetrics/Physical/Metadata/System
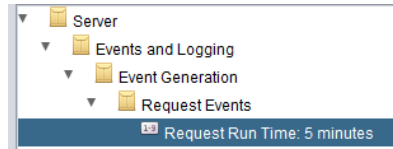
/ClusterSafeCache

/ClusterSafeTriggers

/Helpers

This section lists all triggers and load scripts that have been defined to execute various KPImetrics procedures at regular intervals. The default execution frequencies are listed for each trigger. The load scripts have been created to load and aggregate raw data into processed KPImetrics metrics.

| Trigger [schedule] → Script Name → View name | Description |
|---|---|
| **kpimetricsTrig_00_CheckMetricsActivityDebug [hourly] 'Y',60,06:00:00,20:00:00 → pCheckMetricsActivity →** pGetEmailSubscriptions | This script is used for debugging the DV native metrics and checks for activity on an hourly basis and alerts the configured user whether there was any inactivity. This can be a useful gauge of the health of the DV system. It is configured hourly, but it may make sense to alter the timing to every 2 or more hours depending activity-levels. It is also configured to check between the hours of 6 am to 8 pm daily. It will look at the current timestamp and compare with MAX(starttime) for the collection tables [metrics_requests, metrics_resources_usage] and MAX(logintime) for [metrics_sessions]. It also looks to see if the data transfer jobs have been running by looking at METRICS_JOB_DETAILS. Email integration must be configured on the DV server for this to work. |
| **kpimetricsTrig_01_Cache_ALL_RESOURCES [hourly] → Cache_ALL_RESOURCES →** /System/ALL_RESOURCES → [/services/databases/system/ALL_RESOURCES, l_hostname, l_port] → METRICS_ALL_RESOURCES_STG → EXCEPT_ALL_RESOURCES → METRICS_ALL_RESOURCES | Cache ALL_RESOURCES to make joining with other KPImetrics tables more efficient. Additionally, the METRICS_ALL_RESOURCES table contains a historical record of all "new" resources. It does not delete old ones.<br><br>Insert into METRICS_ALL_RESOURCES select from EXCEPT_ALL_RESOURCES [select from METRICS_ALL_RESOURCES_STG where not exist in METRICS_ALL_RESOURCES] |
| **kpimetricsTrig_02_Cache_ALL_USERS [hourly] → Cache_ALL_USERS →** /System/ALL_USERS → [/services/databases/system/ALL_USERS, l_hostname, l_port] → METRICS_ALL_USERS | Cache ALL_USERS to make joining with other KPImetrics tables more efficient. Additionally, the METRICS_ALL_USERS table contains a historical record of all "new" resources. It does not delete old ones.<br><br>Insert into METRICS_ALL_USERS select from EXCEPT_ALL_USERS [Select from the /System/All_USERS where not exist in METRICS_ALL_USERS] |
| **kpimetricsTrig_03_Cache_LDAP_PERSON [daily] → Cache_LDAP_PERSON →** ./Metadata/System/LDAP_PERSON → /KPImetrics/Physical/Metadata/LDAP/organizationalPerson | Cache LDAP person data once a day. This is not a historical view of users. It gets a new copy each day.<br><br>Insert into METRICS_LDAP_PERSON select from ./Metadata/System/LDAP_PERSON |
| **kpimetricsTrig_04_Cache_CIS_SYSTEM_RESOURCES [hourly] → Cache_CIS_SYSTEM_RESOURCES →** p15MinutesIncrements | Insert 15 min increment rows into METRICS_CIS_SYSTEM_RESOURCES select from METRICS_LOG_MEMORY, METRICS_LOG_IO, METRICS_LOG_DISK, METRICS_CPU_MEMORY_CHECKER. |
| **kpimetricsTrig_05_Cache_CPU_MEMORY_CHECKER [hourly] → Cache_CPU_MEMORY_CHECKER →** ./Metadata/System/CPU_MEMORY_CHECKER → /KPImetrics/Physical/Metadata/ CPUAndMemChecker/CpuAndMemCheckerCjp | Cache the system CPU and memory values.<br><br>Insert into METRICS_CPU_MEMORY_CHECKER select from ./Metadata/System/CPU_MEMORY_CHECKER. |

| | |
|---|---|
| **kpimetricsTrig_06_Cache_LOG_DISK [hourly]** → <br> **Cache_LOG_DISK** → <br> ./Metadata/System/LOG_DISK → <br> /services/databases/system/LOG_DISK | Cache DV system log disk information. <br> Insert into METRICS_LOG_DISK select from <br> ./Metadata/System/LOG_DISK |
| **kpimetricsTrig_07_Cache_LOG_IO [hourly]** → <br> **Cache_LOG_IO** → <br> ./Metadata/System/LOG_IO → <br> /services/databases/system/LOG_IO | Cache DV system IO information. <br> Insert into METRICS_LOG_IO select from <br> ./Metadata/System/LOG_IO |
| **kpimetricsTrig_08_Cache_LOG_MEMORY [hourly]** → <br> **Cache_LOG_MEMORY** → <br> ./Metadata/System/LOG_MEMORY → <br> /services/databases/system/LOG_MEMORY | Cache DV system log memory information. <br> Insert into METRICS_LOG_MEMORY select from <br> ./Metadata/System/LOG_MEMORY |
| **kpimetricsTrig_09_Cache_METRICS_RESOURCES_USAGE_UD [REMOVED]** | DEPRECATED:  Functionality moved into the data_xfer scripts. |
| **kpimetricsTrig_10_Cache_METRICS_SQL_REQUEST [every 30 min]** → <br> **Cache_METRICS_SQL_REQUEST_EXEC** → <br> Cache_METRICS_SQL_REQUEST_GENSQL | Execute the SQL Parser to parse the SQL request (description) originating from metrics_requests_hist.  Retrieve the list of tables and colunns and update the METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE and METRICS_SQL_COLUMNS tables. |
| [No trigger – manual] <br> Cache_METRICS_SQL_REQUEST_EXEC_ADHOC → <br> Cache_METRICS_SQL_REQUEST_GENSQL_ADHOC | Provides a way to manually re-process successfully parsed requests.  This is only necessary if a code-patch has been provided that changes the values that are inserted.  Normally, this will never be executed. |
| **kpimetricsTrig_11_Cache_METRICS_SQL_REQUEST_REPROCESS [daily]** → <br> **Cache_METRICS_SQL_REQUEST_EXEC_REPROCESS** → <br> Cache_METRICS_SQL_REQUEST_GENSQL_REPROCESS | Re-process errors and execute the SQL Parser to parse the SQL request (description) originating from metrics_requests_hist.  Retrieve the list of tables and colunns and update the METRICS_SQL_REQUEST, METRICS_SQL_RESOURCE and METRICS_SQL_COLUMNS tables. |
| **kpimetricsTrig_12_Cache_SYS_CACHES [6 hours]** → <br> **Cache_SYS_CACHES** → <br> ./Metadata/System/SYS_CACHES → <br> /services/databases/system/SYS_CACHES | Cache the system cache status information. <br> Insert into METRICS_SYS_CACHES select from / <br> ./Metadata/System/SYS_CACHES. |
| **kpimetricsTrig_13_Cache_SYS_DATASOURCES [6 hours]** → **Cache_SYS_DATASOURCES** → <br> ./Metadata/System/SYS_DATASOURCES → <br> /services/databases/system/SYS_DATASOURCES | Cache the system datasource information. <br> Insert into METRICS_SYS_DATASOURCES select from ./Metadata/System/SYS_DATASOURCES. |
| **kpimetricsTrig_14_CheckCISWorkflowStatusFail [hourly]** → **pCheckCISWorkflowStatusFail** → <br> [METRICS_CIS_WORKFLOW, pGetEmailSubscriptions] | Check for WORKFLOW_STATUS=F in the METRICS_CIS_WORKFLOW table since the last check.  Each time this procedure is called it puts a marker row in the table with WORKFLOW_NAME=CHECK_WORKFLOW_STATUS <br> Email integration must be configured on the DV server for this to work. |
| **kpimetricsTrig_15_CheckMetricsActivity [hourly] 'N',60,00:00:00.000,23:59:59.999** → **pCheckMetricsActivity** → pGetEmailSubscriptions | This script checks for activity on a daily basis just before midnight and alerts the configured user whether there was any inactivity during the day.  It is configured to look for inactivity between the hours of 12 am – 12 pm in the current day.  This can be a useful gauge of the health of the DV system.  It will look at the current timestamp and compare with MAX(starttime) for the collection tables [metrics_requests, metrics_resources_usage] and MAX(logintime) for [metrics_sessions].  It also looks to see if the data transfer jobs have been running by looking at METRICS_JOB_DETAILS.  Email integration must be configured on the DV server for this to work. |
| **kpimetricsTrig_16_PurgeHistoryData [daily]** → **pPurgeData** | This script purges old data from METRICS tables by executing series of DELETE statements. The purge period for each delete is defined within /Configuration/commonValues script. <br> purgeWorkflowData 120 = 4 months - Purge tables: |

| | |
|---|---|
| | METRICS_CIS_WORKFLOW<br><br>purgeSQLRequests 120 = 4 months - Purge tables: METRICS_SQL_COLUMNS, METRICS_SQL_RESOURCE, METRICS_SQL_REQUEST<br><br>purgeResourceUsage 120 = 4 months - Purge tables: METRICS_CIS_SYSTEM_RESOURCES, METRICS_CPU_MEMORY_CHECKER, METRICS_LOG_DISK, METRICS_LOG_IO, METRICS_LOG_MEMORY, METRICS_SYS_DATASOURCES<br><br>Email integration must be configured on the DV server for this to work. |
| **kpimetricsTrig_17_CheckExceedMemoryPercentRequests [hourly] → pCheckExceedMemoryPercentRequests →** pGetEmailSubscriptions | This procedure queries this procedure in real-time /shared/ASAssets/KPImetrics/Business/Business/requests/pExceededMemoryPercentRequests to generate a list of queries exceeding memory percent per request. It generates an html table containing each of the requests and emails to the subscriber of the event [EXCEEDED_MEMORY]. Email integration must be configured on the DV server for this to work. |
| **kpimetricsTrig_18_CheckLongRunningRequests [system event=requestRunForTooLong] → pCheckLongRunningRequests →** pGetEmailSubscriptions | This procedure queries this procedure in real-time /shared/ASAssets/KPImetrics/Business/requests/ pLongRunningRequests to generate a list of long running requests. It generates an html table containing each of the requests and emails to the subscriber of the event [LONG_RUNNING]. Email integration must be configured on the DV server for this to work. The trigger is activated by the system request event: "Request Run Time". This is set in the Administration Configuration.<br><br> |
| **kpimetricsTrig_19_AllCustom_AccessByUserOvertime [daily] → Cache_AllCustom_AccessByUserOvertime →** /Metadata/Physical/Abstraction/EXCEPT_ACR_AccessByUserOvertime → [METRICS_RESOURCES_USAGE_UD NOT EXISTS METRICS_ACR_ABUOT] | This procedure ivokes "ACR_AccessByUserOvertime" once a day to improve overall query performance for this report. It finds the delta/difference in rows using the original underlying view METRICS_RESOURCES_USAGE_UD and NOT EXISTS in METRICS_ACR_ABUOT. The acronym for the cache table must be short and thus it stands for "All Custom Reports Access By User Over Time". |
| **kpimetricsTrig_20_AllCustom_ActiveResourcesOverPeriodOfTime [daily] → Cache_AllCustom_ActiveResourcesOverPeriodOfTime→** /Metadata/Physical/Abstraction/EXCEPT_ACR_ActiveResourcesOverPeriodOfTime → [METRICS_RESOURCES_USAGE_UD NOT EXISTS METRICS_ACR_ABUOT] | This procedure invokes "EXCEPT_ACR_ActiveResourcesOverPeriodOfTime" once a day to improve overall query performance for this report. It finds the delta/difference in rows using the original underlying view METRICS_RESOURCES_USAGE_UD and NOT EXISTS in METRICS_ACR_AROPOT. The acronym for the cache table must be short and thus it stands for "All Custom Reports Active Resources Over Period Of Time". |
| **kpimetricsTrig_21_AllCustom_ResourceCount_Details [daily] → Cache_AllCustom_ResourceCount_Details→** /AllCustomReports/ResourceCount_DetailsRT → vResourceUsageUD → METRICS_RESOURCES_USAGE_UD | This procedure caches the real-time query "ACR_ResourceCount_DetailsRT" once a day to improve overall query performance for this report. It caches a completely new set of rows to METRICS_ACR_RCD. The acronym for the cache table must be short and thus it stands for "All Custom Reports Resource Count Details". |
| **kpimetricsTrig_22_AllCustom_ResourceCount_Total [daily] → Cache_AllCustom_ResourceCount_Total→** /AllCustomReports/ResourceCount_TotalRT→ vResourceUsageUD → METRICS_RESOURCES_USAGE_UD | This procedure caches the real-time query "ACR_ResourceCount_TotalRT" once a day to improve overall query performance for this report. It caches a completely new set of rows to METRICS_ACR_RCT. The acronym for the cache table must be short and thus it stands for "All Custom Reports Resource Count Total". |

## Metadata System Helpers Scripts

Location:  /shared/ASAssets/KPImetrics/Physical/Metadata/System/Helpers

This section lists all triggers and load scripts that have been defined to execute various KPImetrics procedures at regular intervals. The default execution frequencies are listed for each trigger.  The load scripts have been created to load and aggregate raw data into processed KPImetrics metrics.

| Script Name → Resource name | Description |
|---|---|
| getClusterHostnamesDiff | The purpose of this procedure is to compare the current hostname for the current server with names in the cluster to determine the difference of those names.  This difference will be used to create and use staging tables for the P_METRICS_ALL_TABLES procedure to use.  It requires a unique staging table for each node in the cluster because it performs a table truncate during its processing.<br>hostname1<br>hostname2<br>The difference for the current hostname=1<br>hostname_a<br>hostname_b<br>The difference for the current hostname=a |
| getCurrentTimestamp | This procedure is used to get the current timestamp which was a workaround for a bug in a previous version of DV. |
| getDatasourceConfiguration →<br>  getBasicResourceCursor →<br>  getBasicResourceXSLT | This procedure is used to get the current datasource configuration for various commonValue settings, paths and derived paths.  It invokes a couple of other procedures to get the data source type. |
| p15MinutesIncrements | This procedure returns a cursor of 15 minute increments given a starting timestamp that is passed in.  It generates one full day of 15 increments for a total of 53 records. |
| pGetEmailSubscriptions →<br>  pGetDomainUsers →<br>  [pGetDomainsXSLT,pGetDomainUsersXSLT] | This procedure constructs a list of emails which is used to send error or informational emails based on subscription to DV groups or LDAP groups. |
| pGetSystemInformation | This script is used to get cluster name and server name from /lib/util/getProperties() built in function. This function is used in load scripts and by custom logger to get server name. |
| pStartWorkflow | This script is invoked from each of the load scripts and marks the start of a workflow in the METRICS_CIS_WORKFLOW table. For a given workflow, the script takes workflow name as an input and returns the next workflow start and end time. |
| pEndWorkflow | Like pStartWorkflow script, this script is also called from other load scripts and marks the end of a workflow by updating the METRICS_CIS_WORKFLOW table when a workflow finishes. This script takes workflow name, workflow start and end times, workflow status and number of rows affected as input and updates the METRICS_CIS_WORKFLOW table. |
| pUpdateEventRegLog | This procedure is used to insert/update rows in the METRICS_EVENT_REG_LOG and METRICS_EVENT_REG_LOG_LINEAGE tables. |

## Physical Oracle Triggers and Scripts

Location: /shared/ASAssets/KPImetrics/Physical/Physical/KPI_oracle

This section lists all of the Oracle specific triggers and scripts that have been defined to execute various KPImetrics procedures at regular intervals. The default execution frequencies are listed for each trigger.

| Trigger [schedule] → Script Name → Procedure name | Description |
|---|---|
| **kpimetricsTrig_30_DBMSScheduler [2 hours]** → **P_METRICS_ALL_TABLES_exec** → P_METRICS_ALL_TABLES | This trigger executes the P_METRICS_ALL_TABLES_exec PLSQL procedure to transfer data from the metrics collection tables to the metrics history tables. It inserts a record into the METRICS_JOB_DETAILS table when it starts with a JOB_TABLE_NAME='DBMS_SCHEDULER'. It updates the same row with a STATUS='SUCCESS' or 'FAILURE'. If 'FAILURE' then update the ADDITIONAL_INFO field with the database error. |
| **kpimetricsTrig_31_DBMSSchedulerError [2 hours]** → **pCheckDBMSSchedulerError** → pGetEmailSubscriptions | Send an email if there is a database PLSQL data transfer error that gets generated. Select details from /KPI_oracle/METRICS_JOB_DETAILS. The timing of 2 hours on the odd hour is based on the fact that the DBMS Scheduler trigger runs every 2 hours on the even hour. Therefore, this triggers runs an hour later to allow the PLSQL data transfer script to complete and post any issues or not. <br> Note: This trigger only runs once per cluster because it finds all errors for all nodes if there is a cluster. |
| **kpimetricsTrig_32_DBMSPartitionManager [1 day, 12 am]** → **P_PARTITION_MANAGER_exec** → /KPImetrics/Physical/Metadata/DDL/Oracle/ [03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_ADD, 03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_DROP] | Partition management is required for the metrics history tables when commonValues.partitionNumber and partitionStartDate are configured. The partition manager trigger wakes up once a day at 12 am and determines if a partition needs to be added or dropped. Technically, the only time any actual action will take place is the 1st day of the month unless DV is down at 12 am on the 1st. This is why it is scheduled to run every day to address any downtime. For every day except the 1st, it will simply find no partitions to add or drop based on what is currently in place and how the parttionNumber is configured. For adding a partition, it always looks at the current Year/Month it executes in and calculates the partition for the next month and determines if it exists or not. For dropping a partition, it counts the current number+1 for next month and compares with the partitionNumber to determine if it should drop the oldest partition. |
| [No Trigger. Manually executed if applicable. Oracle only] <br><br> P_REPAIR_METRICS_REQUESTS_HIST | This procedure is used by DV to execute the native PLSQL procedure to repair metrics_requests_hist rows where resourcekind is null and dataservicename is null. <br><br> **Oracle (ONLY) Lineage**: <br> PROCEDURES: <br> /KPImetrics/Physical/Physical/KPI_oracle/ P_REPAIR_METRICS_REQUESTS_HIST_exec → <br> /KPImetrics/Physical/Physical/KPI_oracle/ P_REPAIR_METRICS_REQUESTS_HIST→ <br> /KPImetrics/Physical/Metadata/KPI_oracle/CIS_KPI/ P_REPAIR_METRICS_REQUESTS_HIST |

### Physical SQL Server Triggers and Scripts

Location: /shared/ASAssets/KPImetrics/Physical/Physical/KPI_sqlserver

This section lists all of the SQL Server specific triggers and scripts that have been defined to execute various KPImetrics procedures at regular intervals. The default execution frequencies are listed for each trigger.

| Trigger [schedule] → Script Name → View name | Description |
|---|---|
| **kpimetricsTrig_30_DBMSScheduler [2 hours]** → **P_METRICS_ALL_TABLES_exec** → P_METRICS_ALL_TABLES | This trigger executes the P_METRICS_ALL_TABLES_exec PLSQL procedure to transfer data from the metrics collection tables to the metrics history tables. It inserts a record into the METRICS_JOB_DETAILS table when it starts with a JOB_TABLE_NAME='DBMS_SCHEDULER'. It updates the same row with a STATUS='SUCCESS' or 'FAILURE'. If 'FAILURE' then update the ADDITIONAL_INFO field with the database error. |
| **kpimetricsTrig_31_DBMSSchedulerError [2 hours]** → **pCheckDBMSSchedulerError** → pGetEmailSubscriptions | Send an email if there is a database PLSQL data transfer error that gets generated. Select details from /KPI_oracle/METRICS_JOB_DETAILS. The timing of 2 hours on the odd hour is based on the fact that the DBMS Scheduler trigger runs every 2 hours on the even hour. Therefore, this triggers runs an hour later to allow the PLSQL data transfer script to complete and post any issues or not. Note: This trigger only runs once per cluster because it finds all errors for all nodes if there is a cluster. |
| **kpimetricsTrig_32_DBMSPartitionManager [1 day, 12 am]** → **P_PARTITION_MANAGER_exec** → /KPImetrics/Physical/Metadata/DDL/Oracle/ [03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_ADD, 03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_DROP] | Partition management is required for the metrics history tables when commonValues.partitionNumber and partitionStartDate are configured. The partition manager trigger wakes up once a day at 12 am and determines if a partition needs to be added or dropped. Technically, the only time any actual action will take place is the 1$^{st}$ day of the month unless DV is down at 12 am on the 1$^{st}$. This is why it is scheduled to run every day to address any downtime. For every day except the 1$^{st}$, it will simply find no partitions to add or drop based on what is currently in place and how the parttionNumber is configured. For adding a partition, it always looks at the current Year/Month it executes in and calculates the partition for the next month and determines if it exists or not. For dropping a partition, it counts the current number+1 for next month and compares with the partitionNumber to determine if it should drop the oldest partition. |

# 7  Appendix A – Partitioning Schemes

This section provides information on the various database partition schemes.

## Oracle Partition Scheme

This section describes how Oracle partitioning is utilized. Oracle by far provides the most elegant and easy to implement solution for partitioning. There are very few moving parts and the commands are all inclusive. The following description demonstrates the "Oracle Partition Management Sliding Window Scenario". It shows the SQL statements that get executed for creation, adding and dropping of partitions.

**Step 1.** Configure commonValues
Setup of the \Configuration\commonValues is a key aspect for partitioning. The partitionNumber and partitionStartDate define how many partitions will be initially created and managed and when the first partition start date is configured for. In the example below there will be 3 partitions created starting with 20170501.

>     commonValues.partitionNumber=3

>     commonValues.partitionStartDate='2017-05-01'

Therefore, the history tables will be created as follows:

| Metrics History Table | Partition Name | Partition Rule |
|---|---|---|
| metrics_requests_hist | MR201705 | < 2017-06-01 00:00:00 |
| | MR201706 | < 2017-07-01 00:00:00 |
| | MR201707 | < 2017-08-01 00:00:00 |
| metrics_resources_usage_hist | MRU201705 | < 2017-06-01 00:00:00 |
| | MRU201706 | < 2017-07-01 00:00:00 |
| | MRU201707 | < 2017-08-01 00:00:00 |
| metrics_sessions_hist | MS201705 | < 2017-06-01 00:00:00 |
| | MS201706 | < 2017-07-01 00:00:00 |
| | MS201707 | < 2017-08-01 00:00:00 |

**Step 2.** Create initial history tables, partition strategy and indexes:
[03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables]

Create the history table for the respective tables. For Oracle, the create table statement contains the syntax for creating the partitions. Notice that the PARTITION BY RANGE is used on the "starttime" column. Since the partitionNumber=3 then 3 partitions are created for the initial partition starting with partitionStartDate=2017-05-01. Each partition has a unique name which describes what bucket of data it contains. The partition also contains a rule to compare the data to determine which bucket it goes in.

>     CREATE TABLE "CIS_KPI"."metrics_requests_hist" (<column_list>)

LOB ("description") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW   CHUNK 8192   PCTVERSION 10   NOCACHE   NOLOGGING)

LOB ("message") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW   CHUNK 8192   PCTVERSION 10   NOCACHE   NOLOGGING)

NOCOMPRESS TABLESPACE "METRICS_DATA_HIST"   RESULT_CACHE (MODE DEFAULT)   PCTUSED 0   PCTFREE 10   INITRANS 1   MAXTRANS 255

STORAGE (BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT)

**PARTITION BY RANGE ("starttime")**

(

**PARTITION MR201705 VALUES LESS THAN (TIMESTAMP' 2017-06-01 00:00:00')**

LOGGING NOCOMPRESS TABLESPACE "METRICS_DATA_HIST"

LOB ("description") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0 BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

LOB ("message") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0 BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

PCTFREE 10   INITRANS 1   MAXTRANS 255

STORAGE (INITIAL 8M   NEXT 1M   MAXSIZE UNLIMITED   MINEXTENTS 1   MAXEXTENTS UNLIMITED   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT),

**PARTITION MR201706 VALUES LESS THAN (TIMESTAMP' 2017-07-01 00:00:00')**

LOGGING NOCOMPRESS TABLESPACE "METRICS_DATA_HIST"

LOB ("description") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0 BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

LOB ("message") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0 BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

PCTFREE 10   INITRANS 1   MAXTRANS 255

STORAGE (INITIAL 8M   NEXT 1M   MAXSIZE UNLIMITED   MINEXTENTS 1   MAXEXTENTS UNLIMITED   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT),

**PARTITION MR201707 VALUES LESS THAN (TIMESTAMP' 2017-08-01 00:00:00')**

LOGGING NOCOMPRESS TABLESPACE "METRICS_DATA_HIST"

LOB ("description") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW   CHUNK 8192   RETENTION   NOCACHE   LOGGING

     STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

LOB ("message") STORE AS (TABLESPACE "METRICS_DATA_HIST"   ENABLE STORAGE IN ROW   CHUNK 8192   RETENTION   NOCACHE   LOGGING

     STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

     PCTFREE 10   INITRANS 1   MAXTRANS 255

     STORAGE (INITIAL 8M   NEXT 1M   MAXSIZE UNLIMITED   MINEXTENTS 1   MAXEXTENTS UNLIMITED   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT)

)

NOCACHE NOPARALLEL MONITORING;

Create the indexes for the history tables.  Indexes are created/managed on the main table. Notice that the tablespace can be different for indexes than for the history tables.

CREATE INDEX "mr_hist_rid" ON "CIS_KPI"."metrics_requests_hist" ("requestid", "nodehost", "nodeport") TABLESPACE "METRICS_DATA_IDX";

CREATE INDEX "mr_hist_rid_time" ON "CIS_KPI"."metrics_requests_hist" ("requestid", "starttime", "nodehost", "nodeport") TABLESPACE "METRICS_DATA_IDX";

Repeat all creation operations shown above for "metrics_requests_usage_hist" and "metrics_sessions_hist".

This shows the partition distribution for the history tables.

| table_name | partition_number | num_rows |
|---|---|---|
| "MR201705" | 1 | 0 |
| "MR201706" | 2 | 0 |
| "MR201707" | 3 | 0 |
| "MRU201705" | 1 | 0 |
| "MRU201706" | 2 | 0 |
| "MRU201707" | 3 | 0 |
| "MS201705" | 1 | 0 |
| "MS201706" | 2 | 0 |
| "MS201707" | 3 | 0 |

**Step 3.** Add next month partition:
[03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_ADD]

Notice how a new partition table is created simply with an ALTER TABLE statement. No other operation is required.  Indexes are automatically created.

ALTER TABLE "CIS_KPI"."metrics_requests_hist" ADD

**PARTITION "MR201708" VALUES LESS THAN (TIMESTAMP '2017-09-01 00:00:00')**

LOGGING NOCOMPRESS TABLESPACE "METRICS_DATA"

LOB ("description") STORE AS (TABLESPACE "METRICS_DATA"   ENABLE STORAGE IN ROW   CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

LOB ("message") STORE AS (TABLESPACE "METRICS_DATA"   ENABLE STORAGE IN ROW   CHUNK 8192   RETENTION   NOCACHE   LOGGING

STORAGE (INITIAL 8M   NEXT 1M   MINEXTENTS 1   MAXEXTENTS UNLIMITED   PCTINCREASE 0   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT))

PCTFREE 10   INITRANS 1   MAXTRANS 255

STORAGE (INITIAL 8M   NEXT 1M   MAXSIZE UNLIMITED   MINEXTENTS 1   MAXEXTENTS UNLIMITED   BUFFER_POOL DEFAULT   FLASH_CACHE DEFAULT   CELL_FLASH_CACHE DEFAULT);

Repeat all creation operations shown above for "metrics_requests_usage_hist" and "metrics_sessions_hist".

This shows the partition distribution for the history tables.  Notice how 201708 partitions were added.

| table_name | partition_number | num_rows |
| --- | --- | --- |
| "MR201705" | 1 | 0 |
| "MR201706" | 2 | 0 |
| "MR201707" | 3 | 0 |
| "MR201708" | 4 | 0 |
| "MRU201705" | 1 | 0 |
| "MRU201706" | 2 | 0 |
| "MRU201707" | 3 | 0 |
| "MRU201708" | 4 | 0 |
| "MS201705" | 1 | 0 |
| "MS201706" | 2 | 0 |
| "MS201707" | 3 | 0 |
| "MS201708" | 4 | 0 |

**Step 4.** Insert test rows:
[/shared/ASAssets/KPImetrics/Physical/Metadata/DML/test_insert_metrics_requests_hist]

Notice how there is 1 row in each metrics_requests_hist partitions.

| table_name | partition_number | num_rows |
|---|---|---|
| "MR201705" | 1 | 1 |
| "MR201706" | 2 | 1 |
| "MR201707" | 3 | 1 |
| "MR201708" | 4 | 1 |
| "MRU201705" | 1 | 0 |
| "MRU201706" | 2 | 0 |
| "MRU201707" | 3 | 0 |
| "MRU201708" | 4 | 0 |
| "MS201705" | 1 | 0 |
| "MS201706" | 2 | 0 |
| "MS201707" | 3 | 0 |
| "MS201708" | 4 | 0 |

**Step 5.** Drop the oldest partition.

Test the Drop Scenario by forcing the number of partitions to be 1 less than before with the date incremented by 1 month

        commonValues.partitionNumber=2

        commonValues.partitionStartDate='2017-06-01'

Drop the oldest partition: [03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_DROP]

The strategy for dropping a partition is called the sliding window. It is much more efficient than deleting rows. KPImetrics simply drops the partition. It is very simple.

        ALTER TABLE "CIS_KPI"."metrics_requests_hist" DROP PARTITION "MR201705";

Repeat the same steps for "metrics_resources_usage_hist" and "metrics_sessions_hist":

As shown below, the number of partitions has been reduced from 4 to 3. The Jun 1 2017 boundary representing May 2017 data has been dropped. The oldest partition was dropped. The remaining data shown below is still properly partitioned as expected.

| cluster | nodehost | nodeport | requestid | parentid | sessionid | requesttype | description | starttime | endtime | totaldurati... |
|---|---|---|---|---|---|---|---|---|---|---|
| [NULL] | lsdnecvbe... | 9500 | 2 | 1 | 1 | JDBC | [NULL] | 2017-06-01 07:00... | 2017-06-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 3 | 1 | 1 | JDBC | [NULL] | 2017-07-01 07:00... | 2017-07-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 4 | 1 | 1 | JDBC | [NULL] | 2017-08-01 07:00... | 2017-08-... | 0 |

| table_name | partition_number | num_rows |
|---|---|---|
| "MR201706" | 1 | 1 |
| "MR201707" | 2 | 1 |
| "MR201708" | 3 | 1 |
| "MRU201706" | 1 | 0 |
| "MRU201707" | 2 | 0 |
| "MRU201708" | 3 | 0 |
| "MS201706" | 1 | 0 |
| "MS201707" | 2 | 0 |
| "MS201708" | 3 | 0 |

**Step 6.** Repeat Drop oldest partition.

Now, repeat the drop exercise one more time

      commonValues.partitionNumber=1

      commonValues.partitionStartDate='2017-07-01'

Drop the oldest partition: [03_pqCreateDrop_KPI_Tables_oracle_metrics_history_tables_DROP]

      ALTER TABLE "CIS_KPI"."metrics_requests_hist" DROP PARTITION "MR201706";

As shown below, the number of partitions has been reduced from 3 to 2. The Jul 1 2017 boundary representing June 2017 data has been dropped. The table representing the oldest partition was dropped. The remaining data shown below is still properly partitioned as expected.

| cluster | nodehost | nodeport | requestid | parentid | sessionid | requesttype | description | starttime | endtime | totaldurati... |
|---------|----------|----------|-----------|----------|-----------|-------------|-------------|-----------|---------|---------------|
| [NULL] | lsdnecvbe... | 9500 | 3 | 1 | 1 | JDBC | [NULL] | 2017-07-01 07:00:... | 2017-07-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 4 | 1 | 1 | JDBC | [NULL] | 2017-08-01 07:00:... | 2017-08-... | 0 |

| table_name | partition_number | num_rows |
|------------|------------------|----------|
| "MR201707" | 1 | 1 |
| "MR201708" | 2 | 1 |
| "MRU201707" | 1 | 0 |
| "MRU201708" | 2 | 0 |
| "MS201707" | 1 | 0 |
| "MS201708" | 2 | 0 |

## SQL Server Partition Scheme

This section describes how SQL Server partitioning is utilized. The scenario that is explained is here is referred to as the "SQL Server Partition Management Sliding Window Scenario". It shows the SQL statements that get executed for creation, adding and dropping of partitions.

**Step 1.** Configure commonValues

Setup of the \Configuration\commonValues is a key aspect for partitioning. The partitionNumber and partitionStartDate define how many partitions will be initially created and managed and when the first partition start date is configured for. In the example below there will be 3 partitions created starting with 20170501.

    commonValues.partitionNumber=3

    commonValues.partitionStartDate='2017-05-01'

Therefore the history tables will be created as follows:

| Metrics History Table | Derived Partition Name | Partition Function Rule |
|---|---|---|
| metrics_requests_hist | MR201705 | < 2017-06-01 00:00:00 |
| | MR201706 | < 2017-07-01 00:00:00 |
| | MR201707 | < 2017-08-01 00:00:00 |
| metrics_resources_usage_hist | MRU201705 | < 2017-06-01 00:00:00 |
| | MRU201706 | < 2017-07-01 00:00:00 |
| | MRU201707 | < 2017-08-01 00:00:00 |
| metrics_sessions_hist | MS201705 | < 2017-06-01 00:00:00 |
| | MS201706 | < 2017-07-01 00:00:00 |
| | MS201707 | < 2017-08-01 00:00:00 |

Note that in SQL Server there is no such thing as a partition name. It is simply shown here as the "Derived Partition Name" to describe how the partitions of data are distributed. It is shown fore reporting purposes in the procedure 03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_ROW_DISTRIBUTION.

**Step 2.** Create initial history tables, partition strategy and indexes:
[03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables]

Create the partition function and partition scheme. This partition strategy uses "RANGE RIGHT" as it makes it easier to define a monthly bucket based on the 1st of the month at midnight. All data less than that value is placed in the bucket for the current month. Also notice that the partitioning scheme applies the filegroup. This provides the flexibility of placing each month partition in its own filegroup and potentially disk spindle. However, for the sake of ease of implementation, KPImetrics only implements a single file group for all partitions. These days, the underlying disk architecture is usually hidden anyway with NFS mounted drives.

```
CREATE PARTITION FUNCTION "mr_hist_partition_function" (DATETIME2(3)) AS RANGE RIGHT FOR
VALUES ('2017-06-01 00:00:00', '2017-07-01 00:00:00', '2017-08-01 00:00:00');
```

```
CREATE PARTITION SCHEME "mr_hist_partition_scheme" AS PARTITION "mr_hist_partition_function" ALL
TO ([METRICS_DATA_HIST]);
```

Create the partitioned history table, partitioned archive table and partitioned indexes for the respective tables. The archive table must be created exactly like the history table in order for the SWITCH to take place. This includes the partition scheme and indexes. Notice how the tables are created based on the partition scheme and not the filegroup. The filegroup is actually assigned to the partition scheme as shown previously.

```
CREATE TABLE "dbo"."metrics_requests_hist_arch" (<column_list>) ON
"mr_hist_partition_scheme"("starttime");
```

```
CREATE TABLE "dbo"."metrics_requests_hist" (<column_list>) ON "mr_hist_partition_scheme"("starttime");
```

Create partitioned indexes on history table. The indexes are created on the same partitioning scheme as the tables.

```
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE object_id = object_id(N'metrics_requests_hist') AND
NAME ='mr_hist_rid') CREATE NONCLUSTERED INDEX "mr_hist_rid" ON "dbo"."metrics_requests_hist"
("requestid", "nodehost", "nodeport") ON "mr_hist_partition_scheme"("starttime");
```

```
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE object_id = object_id(N'metrics_requests_hist') AND
NAME ='mr_hist_rid_time') CREATE NONCLUSTERED INDEX "mr_hist_rid_time" ON
"dbo"."metrics_requests_hist" ("requestid", "starttime", "nodehost", "nodeport") ON
"mr_hist_partition_scheme"("starttime");
```

Create partitioned indexes on archive table. The archive table must look exactly like the history table to perform SWITCH.

```
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE object_id = object_id(N'metrics_requests_hist_arch')
AND NAME ='mr_hist_rid_arch') CREATE NONCLUSTERED INDEX "mr_hist_rid_arch" ON
"dbo"."metrics_requests_hist_arch" ("requestid", "nodehost", "nodeport") ON
"mr_hist_partition_scheme"("starttime");
```

```
IF NOT EXISTS(SELECT * FROM sys.indexes WHERE object_id = object_id(N'metrics_requests_hist_arch')
AND NAME ='mr_hist_rid_time_arch') CREATE NONCLUSTERED INDEX "mr_hist_rid_time_arch" ON
"dbo"."metrics_requests_hist_arch" ("requestid", "starttime", "nodehost", "nodeport") ON
"mr_hist_partition_scheme"("starttime");
```

Repeat table and index creation for "metrics_requests_usage_hist" and "metrics_sessions_hist".

This shows the partition distribution for the history tables

| table_name | boundary_definition | partition_number | num_rows |
|---|---|---|---|
| metrics_requests_hist | Jun  1 2017 12:00AM | 1 | 0 |
| metrics_requests_hist | Jul  1 2017 12:00AM | 2 | 0 |
| metrics_requests_hist | Aug 1 2017 12:00AM | 3 | 0 |
| metrics_resources_usage_hist | Jun  1 2017 12:00AM | 1 | 0 |
| metrics_resources_usage_hist | Jul  1 2017 12:00AM | 2 | 0 |
| metrics_resources_usage_hist | Aug 1 2017 12:00AM | 3 | 0 |
| metrics_sessions_hist | Jun  1 2017 12:00AM | 1 | 0 |
| metrics_sessions_hist | Jul  1 2017 12:00AM | 2 | 0 |
| metrics_sessions_hist | Aug 1 2017 12:00AM | 3 | 0 |

**Step 3.** Add next month partition:
[03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_ADD]

Notice how the SPLIT RANGE is used to add the next month range.  If the current month is 2017-07 [Jul] then the boundary is 2017-08.  The next month is 20170-08 [Aug] and the boundary is 2017-09 [Sep].  The following statement alters the scheme with NEXT USED [filegroup] to force the new partition range to take affect.

```
ALTER PARTITION FUNCTION "mr_hist_partition_function"() SPLIT RANGE ('2017-09-01 00:00:00');

ALTER PARTITION SCHEME "mr_hist_partition_scheme" NEXT USED [METRICS_DATA_HIST];

ALTER PARTITION FUNCTION "mru_hist_partition_function"() SPLIT RANGE ('2017-09-01 00:00:00');

ALTER PARTITION SCHEME "mru_hist_partition_scheme" NEXT USED [METRICS_DATA_HIST];

ALTER PARTITION FUNCTION "ms_hist_partition_function"() SPLIT RANGE ('2017-09-01 00:00:00');

ALTER PARTITION SCHEME "ms_hist_partition_scheme" NEXT USED [METRICS_DATA_HIST];
```

**Step 4.** Insert test rows:
[/shared/ASAssets/KPImetrics/Physical/Metadata/DML/test_insert_metrics_requests_hist]

| table_name | boundary_definition | partition_number | num_rows |
|---|---|---|---|
| metrics_requests_hist | Jun  1 2017 12:00AM | 1 | 1 |
| metrics_requests_hist | Jul  1 2017 12:00AM | 2 | 1 |
| metrics_requests_hist | Aug 1 2017 12:00AM | 3 | 1 |
| metrics_requests_hist | Sep 1 2017 12:00AM | 4 | 1 |
| metrics_resources_usage_hist | Jun  1 2017 12:00AM | 1 | 0 |
| metrics_resources_usage_hist | Jul  1 2017 12:00AM | 2 | 0 |
| metrics_resources_usage_hist | Aug 1 2017 12:00AM | 3 | 0 |
| metrics_resources_usage_hist | Sep 1 2017 12:00AM | 4 | 0 |
| metrics_sessions_hist | Jun  1 2017 12:00AM | 1 | 0 |
| metrics_sessions_hist | Jul  1 2017 12:00AM | 2 | 0 |
| metrics_sessions_hist | Aug 1 2017 12:00AM | 3 | 0 |
| metrics_sessions_hist | Sep 1 2017 12:00AM | 4 | 0 |

Notice that there is 1 row in each partition

| cluster | nodehost | nodeport | requestid | parentid | sessionid | requesttype | description | starttime | endtime | totaldurati... |
|---------|----------|----------|-----------|----------|-----------|-------------|-------------|-----------|---------|----------------|
| [NULL] | lsdnecvbe... | 9500 | 1 | 1 | 1 | JDBC | [NULL] | 2017-05-22 07:57:21.407 | 2017-05-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 1 | 1 | 1 | JDBC | [NULL] | 2017-06-22 07:57:21.407 | 2017-06-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 2 | 1 | 1 | JDBC | [NULL] | 2017-07-22 07:57:21.407 | 2017-07-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 3 | 1 | 1 | JDBC | [NULL] | 2017-08-22 07:57:21.407 | 2017-08-... | 0 |

Result rows:1 - 4

**Step 5.** Drop the oldest partition.

Test the Drop Scenario by forcing the number of partitions to be 1 less than before with the date incremented by 1 month

commonValues.partitionNumber=2

commonValues.partitionStartDate='2017-06-01'

Drop the oldest partition:
[03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_DROP]

The strategy for dropping a partition is called the sliding window. It is much more efficient than deleting rows. The idea is to SWITCH the oldest partition with an empty archive table which is configured with the same structure and indexes. This is accomplished with the ALTER TABLE SWITCH PARTITION shown below.

ALTER TABLE "dbo"."metrics_requests_hist" SWITCH PARTITION 1 TO "dbo"."metrics_requests_hist_arch" PARTITION 1;

The next part of the strategy is to perform the ALTER FUNCTION MERGE RANGE to merge the empty partition with the next oldest partition. This will be very fast since the oldest partition designated by the boundary range "2017-06-01 00:00:00" is empty.

ALTER PARTITION FUNCTION mr_hist_partition_function() MERGE RANGE ('2017-06-01 00:00:00');

Finally, the archive table which now contains the rows from the oldest partition of the history table is truncated to make it empty. Truncating is very fast compared to deleting rows. Theoretically, if it is required, the rows could be archived off to off-line storage or a Big Data solution like Hadoop.

TRUNCATE TABLE "dbo"."metrics_requests_hist_arch";

Repeat the same steps for "metrics_resources_usage_hist" and "metrics_sessions_hist":

ALTER TABLE "dbo"."metrics_resources_usage_hist" SWITCH PARTITION 1 TO "dbo"."metrics_resources_usage_hist_arch"  PARTITION 1;

ALTER PARTITION FUNCTION mru_hist_partition_function() MERGE RANGE ('2017-06-01 00:00:00');

TRUNCATE TABLE "dbo"."metrics_resources_usage_hist_arch";

ALTER TABLE "dbo"."metrics_sessions_hist" SWITCH PARTITION 1 TO "dbo"."metrics_sessions_hist_arch" PARTITION 1;

ALTER PARTITION FUNCTION ms_hist_partition_function() MERGE RANGE ('2017-06-01 00:00:00');

TRUNCATE TABLE "dbo"."metrics_sessions_hist_arch";

As shown below, the number of partitions has been reduced from 4 to 3.  The Jun 1 2017 boundary representing May 2017 data has been dropped.  The data in the partition was switched with an empty partitioned archive table.  The data in the archive table was truncated.

The remaining data shown below is still properly partitioned as expected.

| cluster | nodehost | nodeport | requestid | parentid | sessionid | requesttype | description | starttime | endtime | totaldurati... |
|---------|----------|----------|-----------|----------|-----------|-------------|-------------|-----------|---------|----------------|
| [NULL] | lsdnecvbe... | 9500 | 1 | 1 | 1 | JDBC | [NULL] | 2017-06-22 07:57:21.407 | 2017-06-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 2 | 1 | 1 | JDBC | [NULL] | 2017-07-22 07:57:21.407 | 2017-07-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 3 | 1 | 1 | JDBC | [NULL] | 2017-08-22 07:57:21.407 | 2017-08-... | 0 |

| table_name | boundary_definition | partition_number | num_rows |
|------------|---------------------|------------------|----------|
| metrics_requests_hist | Jul  1 2017 12:00AM | 1 | 1 |
| metrics_requests_hist | Aug  1 2017 12:00AM | 2 | 1 |
| metrics_requests_hist | Sep  1 2017 12:00AM | 3 | 1 |
| metrics_resources_usage_hist | Jul  1 2017 12:00AM | 1 | 0 |
| metrics_resources_usage_hist | Aug  1 2017 12:00AM | 2 | 0 |
| metrics_resources_usage_hist | Sep  1 2017 12:00AM | 3 | 0 |
| metrics_sessions_hist | Jul  1 2017 12:00AM | 1 | 0 |
| metrics_sessions_hist | Aug  1 2017 12:00AM | 2 | 0 |
| metrics_sessions_hist | Sep  1 2017 12:00AM | 3 | 0 |

**Step 6.** Repeat Drop oldest partition.

Now, repeat the drop exercise one more time

     commonValues.partitionNumber=1

     commonValues.partitionStartDate='2017-07-01'

Drop the oldest partition:
[03_pqCreateDrop_KPI_Tables_sqlserver_metrics_history_tables_DROP]

ALTER TABLE "dbo"."metrics_requests_hist" SWITCH PARTITION 1 TO "dbo"."metrics_requests_hist_arch" PARTITION 1;

ALTER PARTITION FUNCTION mr_hist_partition_function() MERGE RANGE ('2017-07-01 00:00:00');

TRUNCATE TABLE "dbo"."metrics_requests_hist_arch";

ALTER TABLE "dbo"."metrics_resources_usage_hist" SWITCH PARTITION 1 TO "dbo"."metrics_resources_usage_hist_arch"  PARTITION 1;

ALTER PARTITION FUNCTION mru_hist_partition_function() MERGE RANGE ('2017-07-01 00:00:00');

TRUNCATE TABLE "dbo"."metrics_resources_usage_hist_arch";

ALTER TABLE "dbo"."metrics_sessions_hist" SWITCH PARTITION 1 TO "dbo"."metrics_sessions_hist_arch" PARTITION 1;

ALTER PARTITION FUNCTION ms_hist_partition_function() MERGE RANGE ('2017-07-01 00:00:00');

TRUNCATE TABLE "dbo"."metrics_sessions_hist_arch";

As shown below, the number of partitions has been reduced from 3 to 2. The Jul 1 2017 boundary representing June 2017 data has been dropped. The data in the partition was switched with an empty partitioned archive table. The data in the archive table was truncated.

The remaining data shown below is still properly partitioned as expected.

| cluster | nodehost | nodeport | requestid | parentid | sessionid | requesttype | description | starttime | endtime | totaldurati... |
|---------|----------|----------|-----------|----------|-----------|-------------|-------------|-----------|---------|---------|
| [NULL] | lsdnecvbe... | 9500 | 2 | 1 | 1 | JDBC | [NULL] | 2017-07-22 07:57:21.407 | 2017-07-... | 0 |
| [NULL] | lsdnecvbe... | 9500 | 3 | 1 | 1 | JDBC | [NULL] | 2017-08-22 07:57:21.407 | 2017-08-... | 0 |

| table_name | boundary_definition | partition_number | num_rows |
|------------|---------------------|------------------|----------|
| metrics_requests_hist | Aug 1 2017 12:00AM | 1 | 1 |
| metrics_requests_hist | Sep 1 2017 12:00AM | 2 | 1 |
| metrics_resources_usage_hist | Aug 1 2017 12:00AM | 1 | 0 |
| metrics_resources_usage_hist | Sep 1 2017 12:00AM | 2 | 0 |
| metrics_sessions_hist | Aug 1 2017 12:00AM | 1 | 0 |
| metrics_sessions_hist | Sep 1 2017 12:00AM | 2 | 0 |