



KPI Metrics Metadata Configuration Guide

**An Open Source Asset for use with
TIBCO® Data Virtualization**

TIBCO Software empowers executives, developers, and business users with Fast Data solutions that make the right data available in real time for faster answers, better decisions, and smarter action. Over the past 15 years, thousands of businesses across the globe have relied on TIBCO technology to integrate their applications and ecosystems, analyze their data, and create real-time solutions. Learn how TIBCO turns data—big or small—into differentiation at www.tibco.com.

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



www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304

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

Revision History

Version	Date	Author	Comments
1.0	Aug 30 2019	Mike Tinius	Initial revision

Related Documents

Name	Version
How To Use Utilities.pdf	2019Q301

Supported Versions

Name	Version
TIBCO® Data Virtualization	7.0.8 or later
AS Assets Utilities open source	2019Q301 or later

Table of Contents

1	Introduction	4
	Purpose	4
	Audience	4
	References	4
	Overview	4
2	Requirements	5
3	Configuration.....	6
	Configure Metadata Constants	6
	Configure Trigger	9
4	KPImetrics Metatdata Resources	10
	Configuration Resources.....	10
	Published Resources	10
	KPImetrics Catalog.....	10
	Metadata Table Relationship Diagram.....	12
	Metadata Data Source Tables.....	13
	Metadata Data Source Tables and Procedures for KPI_<database_type>	13
	Metadata System Triggers and Load Scripts.....	16

1 Introduction

Purpose

The purpose of this document is to provide guidance on how configure and use the AS Assets KPI Metadata.

Audience

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

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

References

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

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

Overview

Please review the document “**KPImetrics Overview.pdf**”.

2 Requirements

The following requirements and pre-requisites must be met:

- See requirements section in KPImetrics Configuration Guide vx.yy.pdf.

3 Configuration

Configure Metadata Constants

Background Information:

The procedure “10_pqInsert_Metadata_Tables_METADATA_Constants” is used to configure various constants for a given “project”. A project has a base path which encompasses all of the layer folders and resources.

This procedure “DOES NOT” need to be executed manually. It will be executed each time the trigger “kpimetricsTrig_40_Cache_METADATA_TABLES” executes. The trigger executes Cache_METADATA_TABLES which in turn executes “10_pqInsert_Metadata_Tables_METADATA_Constants”. It does this so that all metadata is kept in synch with the same LOAD_DATE across all of the tables.

Instructions:

- Configure the following /shared/ASAssets/KPImetrics/Physical/Metadata/DDL/Common/10_pqInsert_Metadata_Tables_METADATA_Constants.
- Configure the section “INSERT METADATA_CONST_NAME ROWS”
 - Modify the concatenated string below as needed. Add a row for each "project" name to capture metadata for.
 - PROJECT_NAME: A unique name that will be assigned a unique ID.
 - EXECUTE_FLAG: Y=execute this row. N=do not execute when triggered.
 - ARCHIVE_FLAG: Y=archive rows before processing. N=do not archive.
 - ARCHIVE_PURGE_DAYS: The number of days to purge from the current date.
 - PROJECT_DESC: A description of the project path.
 - Maintain the existing structure with double pipe separating the line and single pipe separating a column.

```
SET projectName = 'TestSpoke';
SET METADATA_CONST_NAME_str = METADATA_CONST_NAME_str ||
PROJECT_NAME      EXECUTE_FLAG      ARCHIVE_FLAG      ARCHIVE_PURGE_DAYS      PROJECT_DESC
'||projectName    'Y'                'N'                30                      'TestSpoke project desc' ||
"; -- This is always the last line
```

- Configure the section “INSERT METADATA_CONST_PATH ROWS”
 - Modify the concatenated string below as needed. Add a row for each base path within the "project" to capture metadata for.
 - Modify projectName, pathSH, pathDS.
 - The variable "pathWS" is not currently supported for web services.
 - Modify the PROJECT_PATH and RESOURCE_TYPES as per your requirements.

- Maintain the existing structure with double pipe separating the line and single pipe separating a column.
- **PROJECT_NAME:** A foreign key reference to **METADATA_CONST_NAME** which provides a unique name that will be assigned a **PROJECT_NAME_ID** that is unique.
- **PROJECT_PATH:** A unique key for this table which drives all of the processing for **Cache_METADATA_TABLES** procedure to load data.
- **RESOURCE_TYPES:** - A comma-separated list of resource types to process.
 - When using pathSH for shared area then [TABLE,PROCEDURE]
 - When using pathDS for /services/databases then [LINK]
 - When using pathWS for /services/webservices then [LINK]
 - NOTE:** Web Services are not currently supported.

```

SET projectName = 'TestSpoke';
SET pathSH = '/shared/00_DataFederation/TestSpoke';
SET pathDS = '/services/databases/PWC/TestSpoke';
SET pathWS = '';
SET METADATA_CONST_PATHS_str = METADATA_CONST_PATHS_str ||
--PROJECT_NAME      PROJECT_PATH      RESOURCE_TYPES
'|'||projectName    '|'||pathSH      '|'||'TABLE,PROCEDURE'  ||
'|'||projectName    '|'||pathDS      '|'||'LINK'              ||
--'|'||projectName  '|'||pathWS      '|'||'LINK'              ||
"; -- This is always the last line

```

- Configure the section “**INSERT METADATA_CONST_LAYERS ROWS**”
 - Modify the concatenated string below as needed. Only modify the layer type and parent path after the standard project path.
 - Modify projectName, pathSH, pathDS.
 - The variable "pathWS" is not currently supported for web services.
 - Modify the **PROJECT_PATH**, **LAYER_TYPE**, **PARENT_PATH** and **GENERATE_LINEAGE** as per your requirements.
 - Maintain the existing structure with double pipe separating the line and single pipe separating a column.
 - **PROJECT_NAME:** A foreign key reference to **METADATA_CONST_NAME** which provides a unique name that will be assigned a **PROJECT_NAME_ID** that is unique.
 - **PROJECT_PATH:** Provides a foreign key back to **META_DRIVER** table.
 - **LAYER_TYPE:** A unique string describing the layer to acquire metadata for.
 - **PARENT_PATH:** The actual path in DV which is associated with the **LAYER_TYPE**.
 - **GENERATE_LINEAGE:** Y=Generate lineage for this layer path. N=Do not generate lineage for this layer path.

```

SET projectName = 'TestSpoke';
SET pathSH = '/shared/00_DataFederation/TestSpoke';
SET pathDS = '/services/databases/PWC/TestSpoke';
SET pathWS = '';
SET METADATA_CONST_LAYERS_str = METADATA_CONST_LAYERS_str ||
--PROJECT_PATH      LAYER_TYPE      PARENT_PATH      GENERATE_LINEAGE
'|'||projectName    '|'||pathSH    '|||'00_DataSource'|'||pathSH||'/00_DataSource'|'||'N'|
'|'||projectName    '|'||pathSH    '|||'01_SourceViewLayer'  '|'||pathSH||'/01_SourceViewLayer'  '|||'N'|
'|'||projectName    '|'||pathSH    '|||'02_ConformingViewLayer'|'||pathSH||'/02_ConformingViewLayer'|'||'N'|
'|'||projectName    '|'||pathSH    '|||'031_CommonEntityModel'|'||pathSH||'/031_CommonModelLayer/031_CommonEntityModel'|'||'N'|

```

```

'|'|projectName ||'|' pathSH||'|' '032_CommonDimensionalModel' ||'|' pathSH||'/03_CommonModelLayer/032_CommonDimensionalModel'
'|'|'N'|
'|'|projectName ||'|' pathSH||'|' '033_CommonAnalyticalModel' ||'|' pathSH||'/03_CommonModelLayer/033_CommonAnalyticalModel' ||'|' 'N'|
'|'|projectName ||'|' pathSH||'|' '034_CommonIntegrationModel' ||'|' pathSH||'/03_CommonModelLayer/034_CommonIntegrationModel' ||'|' 'N'|
'|'|projectName ||'|' pathSH||'|' '041_BusinessDemandModel' ||'|' pathSH||'/04_BusinessDeliveryLayer/041_BusinessDemandModel' ||'|' 'N'|
'|'|projectName ||'|' pathSH||'|' '042_BusinessDemandView' ||'|' pathSH||'/04_BusinessDeliveryLayer/042_BusinessDemandView' ||'|' 'Y'|
'|'|projectName ||'|' pathDS||'|' 'PublishedDS_tutorial' ||'|' pathDS ||'|' 'Y'|
--'|'|projectName ||'|' pathWS||'|' 'PublishedWS_tutorial' ||'|' pathDS ||'|' 'Y'|
"; -- This is always the last line

```

- Configure the section “INSERT METADATA_CONST_VALIDATE ROWS”
 - Modify the concatenated string below as needed.
 - Modify projectName, pathSH, pathDS.
 - The variable "pathWS" is not currently supported for web services.
 - Modify the PROJECT_PATH, LAYER_TYPE, RULE_TYPE and RULE_DESC as per your requirements.
 - Maintain the existing structure with double pipe separating the line and single pipe separating a column.
 - PROJECT_NAME: A foreign key reference to METADATA_CONST_NAME which provides a unique name that will be assigned a PROJECT_NAME_ID that is unique.
 - LAYER_TYPE - A valid layer name found in the table METADATA_CONST_LAYERS.
 - RULE_TYPE - Valid values=[ENFORCE_LAYER|ENFORCE_COLUMN]
 - RULE_DESC - Enforce the rule type.
 - When RULE_TYPE=ENFORCE_COLUMN
 - Enforces which columns must be present in all of the views for a given layer type. Comma-separated list of case-sensitive column names.
 - When RULE_TYPE=ENFORCE_LAYER
 - Enforces which source layer resource can invoke which target layer resource. Comma-separated list of valid LAYER_TYPES.
 - If a resource can invoke another resource in the same layer then add its own layer to the list.

```

SET projectName = 'TestSpoke';
SET pathSH = '/shared/00_DataFederation/TestSpoke';
SET pathDS = '/services/databases/PWC/TestSpoke';
SET pathWS = "";
SET METADATA_CONST_VALIDATE_str = METADATA_CONST_VALIDATE_str ||
--PROJECT_NAME PROJECT_PATH LAYER_TYPE RULE_TYPE RULE_DESC
'|'|projectName ||'|' pathSH ||'|' '01_SourceViewLayer' ||'|' 'ENFORCE_LAYER' ||'|' '00_DataSource' ||'|'
'|'|projectName ||'|' pathSH ||'|' '01_SourceViewLayer' ||'|' 'ENFORCE_COLUMN' ||'|' 'fetchTimeStamp,systemSourceCode' ||'|'
'|'|projectName ||'|' pathSH ||'|' '02_ConformingViewLayer' ||'|' 'ENFORCE_LAYER' ||'|' '01_SourceViewLayer' ||'|'
'|'|projectName ||'|' pathSH ||'|' '031_CommonEntityModel' ||'|' 'ENFORCE_LAYER' ||'|' '02_ConformingViewLayer' ||'|'
'|'|projectName ||'|' pathSH ||'|' '032_CommonDimensionalModel' ||'|' 'ENFORCE_LAYER' ||'|' '02_ConformingViewLayer' ||'|'
'|'|projectName ||'|' pathSH ||'|' '033_CommonAnalyticalModel' ||'|' 'ENFORCE_LAYER' ||'|' '02_ConformingViewLayer' ||'|'
'|'|projectName ||'|' pathSH ||'|' '034_CommonIntegrationModel' ||'|' 'ENFORCE_LAYER' ||'|' '02_ConformingViewLayer' ||'|'
'|'|projectName ||'|' pathSH ||'|' '041_BusinessDemandModel' ||'|' 'ENFORCE_LAYER' ||'|'
'031_CommonEntityModel,032_CommonDimensionalModel,034_CommonIntegrationModel,041_BusinessDemandModel' ||'|'
'|'|projectName ||'|' pathSH ||'|' '042_BusinessDemandView' ||'|' 'ENFORCE_LAYER' ||'|' '041_BusinessDemandModel' ||'|'
'|'|projectName ||'|' pathDS ||'|' 'PublishedDS_tutorial' ||'|' 'ENFORCE_LAYER' ||'|' '042_BusinessDemandView' ||'|'
--'|'|projectName ||'|' pathWS ||'|' 'PublishedWS_tutorial' ||'|' 'ENFORCE_LAYER' ||'|' '042_BusinessDemandView' ||'|'
"; -- This is always the last line

```


Configure Trigger

Enabling triggers starts the processing of KPI metadata data. The trigger “kpimetricsTrig_40_Cache_METADATA_TABLES” is turned off by default. It must be turned on in order to begin the processing of

1. Modify /shared/ASAssets/KPImetrics/Configuration/**defaultTriggersToEnable** and change the trigger kpimetricsTrig_40_Cache_METADATA_TABLES from OFF to ON if you want to capture metadata.
2. When updateTriggers is executed, it will turn on and off the trigger automatically according to how the trigger is set in defaultTriggersToEnable.

4 KPImetrics Metatdata Resources

Configuration Resources

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

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.

KPImetrics Catalog

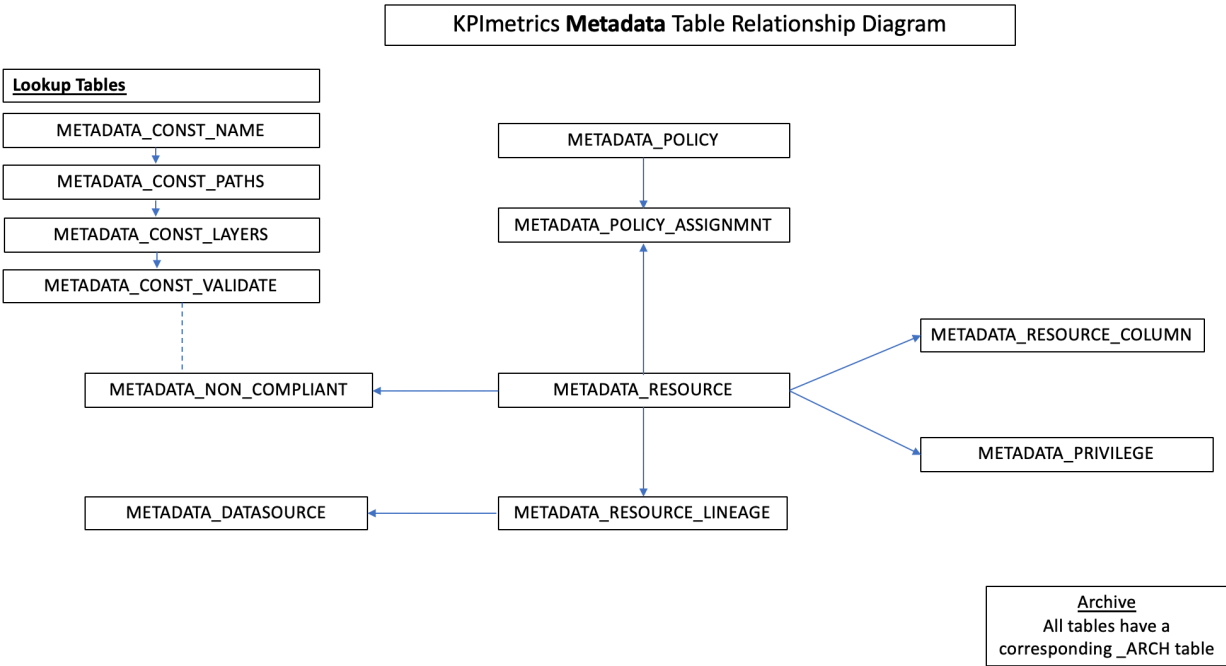
metadata schema

Resource (Resource Type)	Description
reportMetadataDatasource (TABLE)	Report on what datasource a view is using. Lineage from the view to the datasource. This can be joined with vMetadataDatasource to get additional datasource information. It may also be joined with vMetadataResource to get additional resource information.
reportMetadataNonCompliantColumns (TABLE)	Report of the non-compliant columns as configured by METADATA_CONST_VALIDATE.
reportMetadataNonCompliantLayers (TABLE)	Report of the non-compliant layers as configured by METADATA_CONST_VALIDATE.
reportNumResourcesByLayer (TABLE)	Report of the number of resources in each layer.
reportResourceColumns (TABLE)	Report of all the resources and their columns.
vMetadataConstName (TABLE) vMetadataConstNameArch (ARCHIVE TABLE)	Provides a listing of all configured metadata constant name projects. This is the main driver table. It contains the project name, project name id, environment name, execute flag and archive flag.
vMetadataConstPaths (TABLE) vMetadataConstPathsArch (ARCHIVE TABLE)	Provides a listing of all configured metadata constant name projects. This table controls the paths that are connected with the vMetadataConstName view.
vMetadataConstLayers (TABLE) vvMetadataConstLayersArch (ARCHIVE TABLE)	Provides a listing of all of the metadata layer types for each project path.
vMetadataConstValidate (TABLE)	Provides a listing of all of the metadata layer validations for each project path.

vMetadataConstValidateArch (ARCHIVE TABLE)	
vMetadataResource (TABLE) vMetadataResourceArch (ARCHIVE TABLE)	Provides a listing of all of the metadata resources for each project path. This is the core table that provides information about a resource.
vMetadataResourceColumn (TABLE) vMetadataResourceColumnArch (ARCHIVE TABLE)	Provides a listing of all of the metadata resource columns for each resource and project path. This is the core table that provides information about columns and their position.
vMetadataResourceLineage (TABLE) vMetadataResourceLineageArch (ARCHIVE TABLE)	Provides a listing of the lineage for each resource at each layer. Lineage will be different based on which layer is referenced. Each record also references the datasource that is used.
vMetadataDatasource (TABLE) vMetadataDatasourceArch (ARCHIVE TABLE)	Provides a complete reference on the datasources that exist within the project paths referenced by vMetadataConstName.
vMetadataNonCompliant (TABLE) vMetadataNonCompliantArch (ARCHIVE TABLE)	Provides information on all non-compliant resources for all layers.
vMetadataPolicy (TABLE) vMetadataPolicyArch (ARCHIVE TABLE)	Provides information on all policies for RBS [rule-based security] and CBS [column-based security] rows for a given project path.
vMetadataPolicyAssignmnt (TABLE) vMetadataPolicyAssignmntArch (ARCHIVE TABLE)	Provides the policy assignments for each policy.
vMetadataPrivilege (TABLE) vMetadataPrivilegeArch (ARCHIVE TABLE)	Provides a listing of all group and user privileges for each resource for a given project path.

Metadata Table Relationship Diagram

The following provides a table relationship diagram for the database tables and procedures used by KPImetrics Metadata.



Metadata Data Source Tables

The following provides a description for the database tables used by KPImetrics Metadata.

Metadata Data Source Tables and Procedures 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

The following tables have been created in CIS_KPI schema to capture the required data. Each table has a corresponding archive table.

RULES:

- Only one load set of data is stored at any given point in time in the main metadata tables.
- When METADATA_CONST_NAME.ARCHIVE_FLAG=Y then each table is archived to its corresponding archive table.
- Each node in a cluster will contain its own set of metadata rows therefore, NODE_HOST and NODE_PORT are a part of every key. Even though the resource name will be the same, the RESOURCE_ID may be different on any given node. Be sure to do reporting based on a particular NODE_HOST and NODE_PORT.

Table Name	Description
METADATA_CONST_NAME METADATA_CONST_NAME_ARCH	<p>This table contains a unique base project path that drives all of the metadata capture for all of the tables. Only metadata is captured the project paths present in this table. The trigger specified below along with the procedure it invokes is the only mechanism for capturing metadata for all of the metadata tables listed here.</p> <p>LOAD_DATE: The timestamp of the latest metadata load.</p> <p>PROJECT_NAME_ID: A unique sequence id for each project name.</p> <p>PROJECT_NAME: A unique name that will be assigned a PROJECT_NAME_ID that is unique.</p> <p>ENVIRONMENT_NAME: The environment nickname from commonValues.cisServerNickname.</p> <p>EXECUTE_FLAG: Y=execute this row. N=do not execute when triggered.</p> <p>ARCHIVE_FLAG: Y=archive rows before processing. N=do not archive. Note: all rows get deleted each time the trigger executes. Archive is the only way to maintain history.</p> <p>ARCHIVE_PURGE_DAYS: The number of days to purge from the current date.</p> <p>PROJECT_DESC: A description of the project path.</p> <p>RESOURCE_TYPES: TABLE,PROCEDURE - A comma-separated list of resource types to process. Currently only TABLE and PROCEDURE are valid.</p> <p>EXECUTE_STATUS: The status of the latest load. SUCCESS or EXCEPTION which includes the exception message.</p>

	<p>NODE_HOST: Indicates which hostname/node the processing took place on. Multiple hosts/nodes in a cluster.</p> <p>NODE_PORT: Indicates the port of the DV server in which the processing took place on.</p> <p>TRIGGER:</p> <p>/KPImetrics/Physical/Metadata/System/ClusterSafeTriggers/ kpimetricsTrig_40_Cache_METADATA_TABLES→Cache_METADATA_TABLES</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, PROJECT_NAME, NODE_HOST, NODE_PORT</p>
<p>METADATA_CONST_PATHS</p> <p>METADATA_CONST_PATHS_ARCH</p>	<p>This table contains a list of base project paths that drives all of the metadata capture for all of the tables. Only metadata is captured the project paths present in this table.</p> <p>PROJECT_PATH: A unique key for this table which drives all of the processing for Cache_METADATA_TABLES procedure to load data.</p> <p>RESOURCE_TYPES: TABLE,PROCEDURE,LINK - A comma-separated list of resource types to process.</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, PROJECT_PATH, NODE_HOST, NODE_PORT</p>
<p>METADATA_CONST_LAYERS</p> <p>METADATA_CONST_LAYERS_ARCH</p>	<p>This table contains the valid layer types for each project path. A layer type has a corresponding parent path within the project path that it correlates to.</p> <p>PROJECT_PATH: Provides a foreign key back to METADATA_CONST_NAME table.</p> <p>LAYER_TYPE: A unique string describing the layer to acquire metadata for.</p> <p>PARENT_PATH: The actual path in DV which is associated with the LAYER_TYPE.</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, LAYER_TYPE, NODE_HOST, NODE_PORT</p>
<p>METADATA_CONST_VALIDATE</p> <p>METADATA_CONST_VALIDATE_ARCH</p>	<p>This table contains the layer validation rules. The rules provide for enforcing columns within views and which views can invoke views in specific layers.</p> <p>PROJECT_PATH: Provides a foreign key back to METADATA_CONST_NAME table.</p> <p>LAYER_TYPE: A valid layer name found in the table METADATA_CONST_LAYERS.</p> <p>RULE_TYPE: Valid values=[ENFORCE_LAYER ENFORCE_COLUMN]</p> <p>RULE_DESC: Enforce the rule type.</p> <p>When RULE_TYPE=ENFORCE_COLUMN</p> <p>Enforces which columns must be present in all of the views for a given layer type. Comma-separated list of case-sensitive column names.</p> <p>When RULE_TYPE=ENFORCE_LAYER</p> <p>Enforces which source layer resource can invoke which target layer resource. Comma-separated list of valid LAYER_TYPES.</p> <p>If a resource can invoke another resource in the same layer then add its own layer to the list.</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, LAYER_TYPE, RULE_TYPE, NODE_HOST, NODE_PORT</p>
<p>METADATA_RESOURCE</p> <p>METADATA_RESOURCE_ARCH</p>	<p>This is the core table which all other tables reference. This table contains a row for each TABLE and PROCEDURE resource found within the specified PROJECT_PATH in the METADATA_CONST_NAME table.</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, NODE_HOST, NODE_PORT</p>
<p>METADATA_RESOURCE_COLUMN</p> <p>METADATA_RESOURCE_COLUMN_ARCH</p>	<p>This table contains all of the COLUMNS referenced by the RESOURCE_ID in METADATA_RESOURCE.</p> <p>KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, COLUMN_NAME, NODE_HOST, NODE_PORT</p>
<p>METADATA_RESOURCE_LINEAGE</p> <p>METADATA_RESOURCE_LINEAGE_ARCH</p>	<p>This table contains the lineage for each resource in each layer. This will be a very large table.</p>

	KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, LAYER_TYPE, LINEAGE_ORDER, NODE_HOST, NODE_PORT
METADATA_DATASOURCE METADATA_DATASOURCE_ARCH	This table contains the all of the datasource information for a given project path. KEY: LOAD_DATE, PROJECT_NAME_ID, DATASOURCE_ID, NODE_HOST, NODE_PORT
METADATA_NON_COMPLIANT METADATA_NON_COMPLIANT_ARCH	This table contains information on column and layer compliancy based on the METADATA_CONST_VALIDATE rules tables. KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, NON_COMPLIANT_REASON, NODE_HOST, NODE_PORT
METADATA_POLICY METADATA_POLICY_ARCH	This table contains RBS [rule-based security] and CBS [column-based security] rows for a given project path. KEY: LOAD_DATE, PROJECT_NAME_ID, POLICY_ID, NODE_HOST, NODE_PORT
METADATA_POLICY_ASSIGNMNT METADATA_POLICY_ASSIGNMNT_ARCH	This table contains the assignments for a policy. KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, POLICY_ID, NODE_HOST, NODE_PORT
METADATA_PRIVILEGE METADATA_PRIVILEGE_ARCH	This table contains the assigned privileges for all of the resources in a given project path. KEY: LOAD_DATE, PROJECT_NAME_ID, RESOURCE_ID, NAME, NAME_TYPE, DOMAIN_NAME, USER_NAME, NODE_HOST, NODE_PORT

Metadata System Triggers and Load Scripts

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

/ClusterSafeCache

/ClusterSafeTriggers

/Helpers

This section provides a quick summary of all triggers, their schedules and how they execute in a cluster.

Trigger Name	Trigger Schedule	Trigger Period	Cluster execution
kpimetricsTrig_40_Cache_METADATA_TABLES	2:00 AM	1 day	all nodes

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 metadata.

Trigger [schedule] → Script Name → View name	Description
Schedule: [1 day, 2:00 am] kpimetricsTrig_40_Cache_METADATA_TABLES → pRebuildIndexesAdminInterface → /shared/ASAssets/KPImetrics/Physical/Metadata/System/ClusterSafeCache/Cache_METADATA_TABLES	<p>This trigger executes the Cache_METADATA_TABLES procedure. This procedure is used to capture all the metadata for all of the metadata tables.</p> <p>Exceptions: Emails will be sent if there are exceptions. Review the following view (table) for issues: /services/databases/ASAssets/KPImetrics/workflow/vCISWorkflowStatus</p>