MICRO FOCUS

Silk Central 21.1

The Reporting Data Mart

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2022-01-27

Contents

Overview

The Silk Central reporting data mart makes it easy to access data for reporting purposes. It moves data from the production tables into dedicated views which should be used for creating advanced reports. The advantages include:

- Clear naming of tables and views, allowing you to quickly locate the data you are looking for.
- Pre-processed data, giving you the possibility to access aggregated data without having to calculate it yourself.
- Performance improvement, as reports can use much simpler and faster SQL queries.
- Less dependency on production database load, which also improves performance and removes load spikes.

The current version of the data mart covers the results area. Further areas for reporting will be added to the data mart in future releases. The following tables and views are currently available:

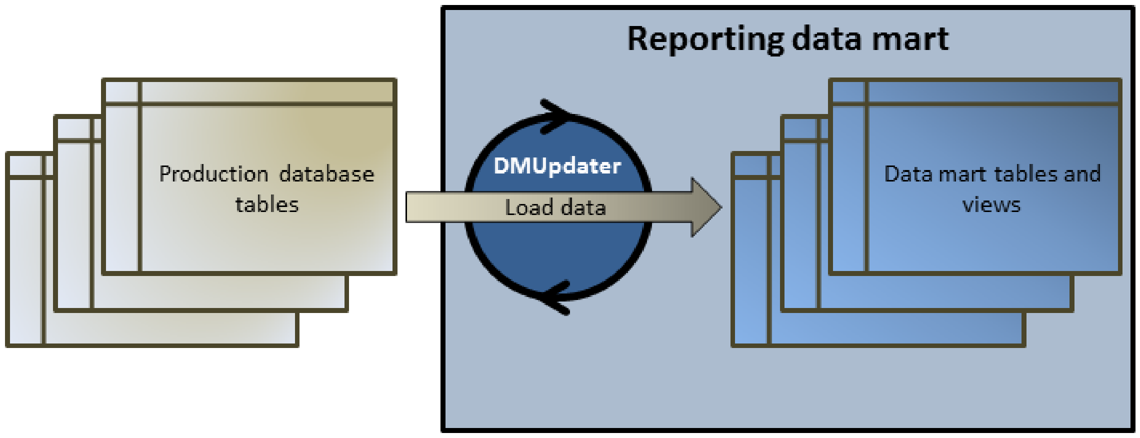
- The DM_TestStatus table is the basis for status-related views.
- The RV_TestStatusExtended view provides detailed information for a certain test execution.
- The RV_LatestTestStatus view provides status and extended information on the latest test run of a test within the context of a specific execution plan and build.
- The RV_LatestFinishedTestStatus view provides status and extended information on the latest test run of a test within the context of a specific execution plan and build. This view provides similar information as the RV_LatestTestStatus view, but only for test runs within finished execution plan runs.
- The RV_MaxTestRunID view is a helper to retrieve the latest test run ID for every test, execution plan, and build combination.
- The RV_MaxFinishedTestRunID view is a helper to retrieve the latest test run ID for every test, execution plan, and build combination. This view provides similar information as the RV_MaxTestRunID view, but only for test runs from finished execution plan runs.
- The RV_TestingCycleStatus view provides status information for testing cycles.
- The RV_ExecutionPlanStatusPerBuild view retrieves the latest test status sums for every execution plan in context of builds.
- The RV_EPFinishedStatusPerBuild view retrieves the latest test status sums for every execution plan in context of builds. This view provides similar information as the RV_ExecutionPlanStatusPerBuild view, but only for test runs from finished execution plan runs.
- The RV_ExecutionPlanStatusRollup view retrieves the sums for passed, failed, and not-executed tests per execution plan or folder in context of a build.
- The RV_EPFinishedStatusRollup view retrieves the sums for passed, failed, and not-executed tests per execution plan or folder in context of a build. This view provides similar information as the RV_ExecutionPlanStatusRollup view, but only for test runs from finished execution plan runs.
- The RV_ConfigurationSuiteStatus view lists the status counts for all configuration suites and configurations per build.
- The RV_ConfigSuiteFinishedStatus view lists the status counts for all configuration suites and configurations per build. This view provides similar information as the RV_ConfigurationSuiteStatus view, but only for test runs from finished execution plan runs.

You can download a .zip file with detailed information about the database schema of Silk Central. In the menu, click **Help** > **Documentation** and then click **Silk Central Database Schema** to download the .zip file.

Architecture

Data is periodically extracted in the background from the production database tables and loaded into the data mart tables and views for easy and fast querying. If the load on the database is not too high, this data is usually available within less than a minute after any changes have been committed. If you are logged in as a system administrator, you can check the current state of the data loading process by navigating to http://<server>:<port>/sctm/check/db and checking the **DM_TestStatus Table**.

Note: If you are updating from a Silk Central version that did not include the data mart (before version 13.0), the data mart tables and views are initially filled with data from the production system. Depending on your database size, this process can take some time. Once this process has completed, you can access the data.



How to Create Reports with the Data Mart

The following examples demonstrate how to create useful reports with the data mart views.

Writing Data Mart Queries

1. In the menu, click **Reports** > **Details View**.
2. In the **Reports** tree, select the folder in which you want the new report to display.
   This determines where the report is stored in the directory tree.
3. Click on the toolbar. The **Create New Report** dialog box opens.
4. Type the name of the new report.
   This is the name that is displayed in the **Reports** tree.
5. Check the **Share this report with other users** check box if you want to make this report available to other users.
6. Type a description of the report in the **Description** field.
7. Click **Advanced Query** to open the **Report data query** field. Insert previously written code or write new code directly in the field.
   The **Insert placeholder** list assists you in editing the SQL queries with predefined function placeholders. For details, see *SQL Functions for Custom Reports*.
   Note: If you manually edit SQL code for the query, click **Check SQL** to confirm your work.
8. Click **Finish** to save your settings.

Reliability of Tests in an Execution Plan

Problem

In a continuous integration environment tests are ideally executed at least once per day for testing the daily build and ensuring the quality of your application under test. To understand how reliable your test set is for measuring the quality of your AUT it is inevitable to sometimes have a look at how the results change over time. For example you might have tests in your test set that frequently change status, therefore being no real measure for quality.

Solution

Use the data mart view *RV_TestStatusExtended* to create a report that lists the results for a specific test in the context of a specific execution plan. This allows you to see how this test's results have changed over time. For convenience, we will narrow the list of results down to those related to tagged builds, thus looking at specific milestone builds of the application under test only. This report collects test result data for tests in the context of execution plans and builds. In the following query we:

- Select the columns we want to display from this view.
- Narrow the result down by the ID of the test we want to investigate and the ID of the execution plan in which the test belongs.
- Add a constraint to consider tagged builds only.

```
SELECT TestName, ExecutionPlanName, VersionName, BuildName, TestRunID,
 PassedCount, FailedCount, NotExecutedCount
```

```
FROM RV_TestStatusExtended
WHERE TestID = ${TESTID|1|Test ID} AND ExecutionPlanID = ${EXECUTIONPLANID|1|Execution Plan
ID} AND BuildIsTagged = 1
ORDER BY BuildOrderNumber
```

The result of the SQL query are all test runs for the selected test within the selected execution plan. In the following example you can see that the test was re-run against build 579_Drop2:

TestName	Execution PlanName	Version Name	BuildName	TestRunID	Passed Count	FailedCount	NotExecute dCount
UI Tests	EN|SQL2012|IE9|IIS	3.0	579_Drop02	7741797	59	5	0
UI Tests	EN|SQL2012|IE9|IIS	3.0	579_Drop02	7745078	63	1	0
UI Tests	EN|SQL2012|IE9|IIS	3.0	593_Drop03	7787437	63	1	0
UI Tests	EN|SQL2012|IE9|IIS	3.0	605_Drop04	7848720	63	1	0

All Failed Tests in an Execution Folder

Problem

Typically all execution plans are structured in a folder hierarchy which identifies the different areas or purposes to which the execution plans and their tests are related. The execution plans are triggered on a regular basis in a continuous integration environment, or occasionally over the release time frame, resulting in nice execution statistics – unfortunately for each single execution plan only.

However, sometimes you need an overall information to know how all your tests perform for a specific area or purpose to identify where the weaknesses are.

Solution

Use the data mart view *RV_LatestTestStatus* to create a report that returns a list of all failed tests for a specific execution planning hierarchy level.

The following query selects failed tests within an execution planning folder with context information like execution plan name and build name:

```
SELECT TestID, TestName, ExecutionPlanID, ExecutionPlanName, BuildName
FROM RV_LatestTestStatus lts
INNER JOIN TM_ExecTreePaths ON lts.ExecutionPlanID = TM_ExecTreePaths.NodeID_pk_fk
WHERE TM_ExecTreePaths.ParentNodeID_pk_fk = ${executionFolderID|2179|Execution Folder ID}
  AND StatusID = 2
ORDER BY TestName
```

The query does the following:

- Uses the view RV_LatestTestStatus for retrieving the latest test run result.
- Includes the execution tree hierarchy (TM_ExecTreePaths) to be able to query all tests from all the execution plans within the hierarchy.
- Uses the top level folder ID from where the analysis should be started as ParentNodeID_pk_fk.

- Includes only failed tests (StatusID = 2).

The StatusID can be looked up in the table TM_TestDefStatusNames.

The result of the SQL query are all tests in the selected execution folder for which the last run failed.

TestID	TestName	ExecutionPlanID	ExecutionPlanName	BuildName
14073	JUnitTestPackage	2184	CI Testing	352
14107	Volatile Tests	2191	Volatile Test	352

Testing Cycle Status

Problem

Testing cycles can be complex objects as they contain information about manual testers, tests, different builds and versions of products, and maybe even configurations. To not lose track it is important to find answers to questions like:

- How many tests have been finished?
- How many of them passed or failed per build?
- Are my manual testers still busy or can they do additional work?

Solution

Use the data mart view *RV_TestingCycleStatus* to create a report that shows the status of a testing cycle per tester and build that will give you an overview of how many tests are passed, failed, not executed grouped by manual tester, configuration and build.

```
SELECT BuildName, TesterLogin, TesterExecutionName,
  PassedCount, FailedCount, NotExecutedCount
FROM RV_TestingCycleStatus
WHERE TestingCycleID = ${testingCycleID|3|Testing Cycle ID}
ORDER BY BuildOrderNumber, TesterLogin
```

The query does the following:

- Uses the view RV_TestingCycleStatus as data source, as it contains BuildName, TesterLogin and TesterExecutionName, which is the generated name reflecting tester, configuration and test.
- Limits the data to the testing cycle ID that you are interested in.

The result of the SQL query shows the status per build and tester.

BuildName	TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecuted Count
352		No specific tester (Test Assets)	0	0	1
351	admin	admin (English|SQL2008|FF|Tomcat - Test Assets)	0	1	0
352	admin	admin (English|SQL2008|FF|Tomcat - Test Assets)	0	0	1

BuildName	TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecuted Count
352	gmazzuchelli	gmazzuchelli (English|Oracle10g|IE8|Tomcat - Test Assets)	0	1	1
352	jallen	jallen (German|Oracle11g|FF|Tomcat - Test Assets)	1	1	0
352	smiller	smiller (German|SQL2008|IE8|IIS - Test Assets)	1	1	0

For unassigned tests a "no specific tester" group is created with empty values for TesterLogin, TesterFirstName, and TesterLastName.

In case you just want to see how your test cycle is doing based on the performance of your manual testers, a slight variation of the query will help:

```
SELECT TesterLogin, TesterExecutionName, SUM(PassedCount) PassedCount,
  SUM(FailedCount) FailedCount, SUM(NotExecutedCount) NotExecutedCount
FROM RV_TestingCycleStatus
WHERE TestingCycleID = ${testingCycleID|3|Testing Cycle ID}
GROUP BY TesterLogin, TesterExecutionName
ORDER BY TesterLogin
```

The query is extended by:

- GROUP BY TesterLogin, TesterExecutionName for denoting the remaining columns.
- SUM() to the counters for aggregating the figures.

TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecutedCount
	No specific tester (Test Assets)	0	0	1
admin	admin (English|SQL2008|FF|Tomcat - Test Assets)	0	1	1
gmazzuchelli	gmazzuchelli (English|Oracle10g|IE8|Tomcat - Test Assets)	0	1	1
jallen	jallen (German|Oracle11g|FF|Tomcat - Test Assets)	1	1	0
smiller	smiller (German|SQL2008|IE8|IIS - Test Assets)	1	1	0

Execution Tree Status

Problem

It is a common practice to have execution plans in a hierarchical structure that represents different testing areas or purposes. In some cases, for example for knowing the test status and therefore the quality of an area or purpose, you will want to know the overall passed, failed, and not executed count.

Solution

Use the data mart view *RV_ExecutionPlanStatusRollup* to create a report that returns the passed, failed, and not executed counts grouped by build for a specific execution planning folder.

```
SELECT BuildName, PassedCount, FailedCount, NotExecutedCount
FROM RV_ExecutionPlanStatusRollup
WHERE ExecutionFolderID = ${executionPlanID|43|Execution Plan ID}
```

The query does the following:

- Selects BuildName and the status counts from the RV_ExecutionPlanStatusRollup view.
- Specifies the top-level folder you want the status from (ExecutionFolderID).

The result of the SQL query shows the status of your test runs in all execution plans of the selected folder, aggregated per build.

BuildName	PassedCount	FailedCount	NotExecutedCount
351	0	0	2
352	15	7	1

If you are interested in more details, for example the status counts for each execution plan within the selected hierarchy, you can use the data mart view *RV_ExecutionPlanStatusPerBuild*:

```
SELECT eps.BuildName, eps.ExecutionPlanID, SUM(eps.PassedCount) PassedCount,
 SUM(eps.FailedCount) FailedCount, SUM(eps.NotExecutedCount) NotExecutedCount
FROM RV_ExecutionPlanStatusPerBuild eps
INNER JOIN TM_ExecTreePaths etp ON eps.ExecutionPlanID = etp.NodeID_pk_fk
WHERE etp.ParentNodeID_pk_fk = ${execFolderID|44|Execution Folder ID}
GROUP BY eps.ExecutionPlanID, eps.BuildOrderNumber, eps.BuildName
ORDER BY eps.BuildOrderNumber, eps.ExecutionPlanID
```

The query does the following:

- Uses the RV_ExecutionPlanStatusPerBuild view to access execution-plan specific data (ExecutionPlanID and ExecutionPlanName). The previously used RV_ExecutionPlanStatusRollup view contains pre-aggregated data (summed up data), which is not suitable for the purpose here as you would get results not only for execution plans but for the folder nodes as well.
- Selects all nodes within a specific folder with a JOIN of the TM_ExecTreePath table to bring in hierarchy information.
- Specifies the top-level folder with ExecutionFolderID. As the table TM_ExecutionTreePaths also contains a self-reference for every execution plan, you could run this query with an execution plan ID for ParentNodeID_pk_fk too, which would return the rows for the specific execution plan.
- Adds ORDER BY BuildOrderNumber and ExecutionPlanID to get a nicely ordered result, showing the oldest builds and their execution plans first.

The result of the SQL query shows the status of your test runs in all execution plans of the selected folder.

BuildName	ExecutionPlanID	PassedCount	FailedCount	NotExecutedCount
351	2307	0	0	2
352	2184	11	2	0
352	2185	0	3	0
352	2186	2	1	0
352	2187	1	0	0
352	2191	0	1	0
352	2307	1	0	1

Configuration Suite Status

Problem

Configuration suites allow you to execute the same set of tests against multiple configurations, for example multiple browsers or operating systems. To be able to make reasonable statements related to quality and reliability of your application under test you will want to keep track of the results for each individual configuration.

Solution

Use the data mart view *RV_ConfigurationSuiteStatus* to create a report that returns the passed, failed, and not executed counts for each configuration per build.

```
SELECT BuildName, ConfigurationName, PassedCount, FailedCount, NotExecutedCount
FROM RV_ConfigurationSuiteStatus
WHERE ConfigurationSuiteID = ${configSuiteID|97|Configuration Suite ID}
ORDER BY BuildOrderNumber, ConfigurationName
```

The query does the following:

- Retrieves the status counts per build of test runs from the RV_ConfigurationSuiteStatus view.
- Narrows the results down to the configuration suite (ConfigurationSuiteID).

The result of the SQL query shows the status of your test runs for each configuration.

BuildName	ConfigurationName	PassedCount	FailedCount	NotExecutedCount
350	Chrome	0	1	0
350	Firefox	0	1	0
350	Internet Explorer	0	1	0
351	Chrome	1	0	0
351	Firefox	1	0	0
351	Internet Explorer	0	1	0
352	Chrome	1	0	0
352	Firefox	1	0	0
352	Internet Explorer	1	0	0

In this example, we use the ID of the configuration suite to get all configurations. It is also possible to restrict the result to specific builds, in which case you would have to include BuildID, BuildName, or BuildOrderNumber in the where clause.

Note: The view *RV_ConfigurationSuiteStatus* only contains aggregated status counts without any test-specific data. To retrieve additional test-specific data, you can use, for example, the view *RV_LatestTestStatus*.

Troubleshooting

Wrong or Missing Data

Problem

When querying data from a data mart table or view, the listed results are not up to date or missing.

Resolution

The data mart tables and views are updated periodically in the background, but not in real time. Due to this, it can take a few seconds up to a few minutes for the data to be loaded into the data mart tables. If your system is running a heavy load, this influences the performance of the background process which is loading the data. The reason is that other processes are prioritized higher and may temporarily block the DataMartUpdater background job. Run your query again later to retrieve updated data.

If you are logged in as a system administrator, you can check the current state of the data loading process by navigating to http://<server>:<port>/sctm/check/db and checking the **DM_TestStatus Table**.

Note: Tests and depending test runs are removed from the data mart if a test is deleted. This also applies to deleted tests due to cleaning up test packages.

The Data Mart Slows Down the System

Problem

Since running the data mart, the system's overall performance seems to be poorer or behaves inconsistently.

Resolution

While this should not happen, you can turn off the data mart to check if this actually resolves your issues:

1. On the **Instance Administration** page, stop the application server of the instance that you want to modify.
2. Open the TMAppServerHomeConf.xml file with a text editor.The default path for this file is C:\Program Files (x86)\Silk\Silk Central 21.1\instance_<instance number>_<instance name>\Conf\AppServer on the application server.
3. Locate the Config/DataMart/Enabled XML tag and set the value to false.
4. Save and close the XML file.
5. Restart the application server.

Reference: Data Mart Tables and Views

The following data mart tables and views are available for easy and fast reporting.

DM_TestStatus

The DM_TestStatus table is the basis for status-related views.

The other data mart views usually provide easier access to detailed data, as this table does not provide direct access to information like the name of a test. The key of this table is the combination of the columns TestID, ExecutionPlanID, BuildID, and TestRunID.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
TestRunID	Identifier of the test run.
ExecutionRunID	Identifies in which execution run this result was generated.
StatusID	Status of this test run (see TM_TestDefStatusNames).
EDRStatusID	Status of this execution run. For additional information, see TM_TestDefStatusNames. For example, status 7 = pending manual run, status 10 = pending automated run.
ReasonID	Reason for the status of this test run (see TM_ResultStatusReasons). Can be null.
PassedCount	Sum of all passed tests, which is 0 or 1 for common tests and can be more for package test roots.
FailedCount	Sum of all failed tests, which is 0 or 1 for common tests and can be more for package test roots.
NotExecutedCount	Sum of all not-executed tests, which is 0 or 1 for common tests and can be more for package test roots.
ProjectID	ID of the project that this row belongs to.
TestStartTime	Time when the test run started (UTC).
ExecutionStartTime	Time when the execution run started (UTC).
TestDurationInMilliseconds	Duration of the test run in milliseconds.
IsBlocked	Flags the test run as blocked/unblocked
DbChangedAt	Time when this row was last updated by the reporting data mart.

RV_TestStatusExtended

The RV_TestStatusExtended view provides detailed information for a certain test execution.

This view contains all test runs, in contrast to the view *RV_LatestTestStatus* which contains only the latest test run of a test within the context of an execution plan and a certain build. You can use this view for example to create a *report that lists all test runs of your tagged builds*. The key of this table is the combination of the columns TestID, ExecutionPlanID, BuildID, and TestRunID.

Note: Tests and depending test runs are removed from the data mart if a test is deleted. This also applies to deleted tests due to cleaning up test packages.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
TestRunID	Identifier of the test run.
ExecutionRunID	Identifies in which execution run this result was generated.
StatusID	Status of this test run (see TM_TestDefStatusNames).
ReasonID	Reason for the status of this test run (see TM_ResultStatusReasons). Can be null.
PassedCount	Sum of all passed tests, which is 0 or 1 for common tests and can be more for package test roots.
FailedCount	Sum of all failed tests, which is 0 or 1 for common tests and can be more for package test roots.
NotExecutedCount	Sum of all not-executed tests, which is 0 or 1 for common tests and can be more for package test roots.
ProjectID	ID of the project that this row belongs to.
TestStartTime	Time when the test run started (UTC).
ExecutionStartTime	Time when the execution run started (UTC).
TestDurationInMilliseconds	Duration of the test run in milliseconds.
IsBlocked	Flags the test run as blocked/unblocked
DbChangedAt	Time when this row was last updated by the reporting data mart.
TestName	Name of the test.
TestDescription	Description of the test.
TestParentID	ID of the test's parent.
PlannedTimeInMinutes	Time planned for this test in minutes.
Reason	Name of the reason. Can contain reasons that have been deleted in the meantime.
ExecutionPlanName	Name of the execution plan.

Row	Description
ExecutionPlanDescription	Description of the execution plan.
ExecutionParentFolderID	ID of the execution plan's parent.
Priority	Priority of the execution plan: 0 = Low, 1 = Medium, 2 = High.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_LatestTestStatus

The RV_LatestTestStatus view provides status and extended information on the latest test run of a test within the context of a specific execution plan and build.

Use the *RV_TestStatusExtended* view to retrieve information about all test runs. You can use this view to create a *report that lists all failed tests in an execution folder*. The key of this table is the combination of the columns TestID, ExecutionPlanID, BuildID, and TestRunID.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
TestRunID	Identifier of the test run.
ExecutionRunID	Identifies in which execution run this result was generated.
StatusID	Status of this test run (see TM_TestDefStatusNames).
ReasonID	Reason for the status of this test run (see TM_ResultStatusReasons). Can be null.
PassedCount	Sum of all passed tests, which is 0 or 1 for common tests and can be more for package test roots.
FailedCount	Sum of all failed tests, which is 0 or 1 for common tests and can be more for package test roots.

Row	Description
NotExecutedCount	Sum of all not-executed tests, which is 0 or 1 for common tests and can be more for package test roots.
ProjectID	ID of the project that this row belongs to.
TestStartTime	Time when the test run started (UTC).
ExecutionStartTime	Time when the execution run started (UTC).
TestDurationInMilliseconds	Duration of the test run in milliseconds.
IsBlocked	Flags the test run as blocked/unblocked
DbChangedAt	Time when this row was last updated by the reporting data mart.
TestName	Name of the test.
TestDescription	Description of the test.
TestParentID	ID of the test's parent.
PlannedTimeInMinutes	Time planned for this test in minutes.
Reason	Name of the reason. Can contain reasons that have been deleted in the meantime.
ExecutionPlanName	Name of the execution plan.
ExecutionPlanDescription	Description of the execution plan.
ExecutionParentFolderID	ID of the execution plan's parent.
Priority	Priority of the execution plan: 0 = Low, 1 = Medium, 2 = High.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_LatestFinishedTestStatus

The RV_LatestFinishedTestStatus view provides status and extended information on the latest test run of a test within the context of a specific execution plan and build. This view provides similar information as the RV_LatestTestStatus view, but only for test runs within finished execution plan runs.

Use this view to create reports that ignore currently running execution plans.

Use the *RV_TestStatusExtended* view to retrieve information about all test runs. The key of this table is the combination of the columns TestID, ExecutionPlanID, BuildID, and TestRunID.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
TestRunID	Identifier of the test run.
ExecutionRunID	Identifies in which execution run this result was generated.
StatusID	Status of this test run. For additional information, see the TM_TestDefStatusNames table in the database schema of Silk Central.
EDRStatusID	Status of this execution run. For example, status 7 = pending manual run, status 10 = pending automated run. For additional information, see the TM_TestDefStatusNames table in the database schema of Silk Central.
ReasonID	Reason for the status of this test run. Can be null. For additional information, see the TM_ResultStatusReasons table in the database schema of Silk Central.
PassedCount	Sum of all passed tests, which is 0 or 1 for common tests and can be more for package test roots.
FailedCount	Sum of all failed tests, which is 0 or 1 for common tests and can be more for package test roots.
NotExecutedCount	Sum of all not-executed tests, which is 0 or 1 for common tests and can be more for package test roots.
ProjectID	ID of the project that this row belongs to.
TestStartTime	Time when the test run started (UTC).
ExecutionStartTime	Time when the execution run started (UTC).
TestDurationInMilliseconds	Duration of the test run in milliseconds.
IsBlocked	Flags the test run as blocked/unblocked
DbChangedAt	Time when this row was last updated by the reporting data mart.
TestName	Name of the test.
TestDescription	Description of the test.
TestParentID	ID of the test's parent.
PlannedTimeInMinutes	Time planned for this test in minutes.

Row	Description
Reason	Name of the reason. Can contain reasons that have been deleted in the meantime.
ExecutionPlanName	Name of the execution plan.
ExecutionPlanDescription	Description of the execution plan.
ExecutionParentFolderID	ID of the execution plan's parent.
Priority	Priority of the execution plan: 0 = Low, 1 = Medium, 2 = High.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_MaxTestRunID

The RV_MaxTestRunID view is a helper to retrieve the latest test run ID for every test, execution plan, and build combination.

The key of this table is the combination of the columns TestID, ExecutionPlanID and BuildID.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
MaxTestRunID	Identifies the latest test run for the test in context of the execution plan and build.

RV_MaxFinishedTestRunID

The RV_MaxFinishedTestRunID view is a helper to retrieve the latest test run ID for every test, execution plan, and build combination. This view provides similar information as the RV_MaxTestRunID view, but only for test runs from finished execution plan runs.

Use this view to create reports that ignore currently running execution plans.

The key of this table is the combination of the columns TestID, ExecutionPlanID, and BuildID.

Row	Description
TestID	Identifier of the test.
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
MaxTestRunID	Identifies the latest test run for the test in context of the execution plan and build.

RV_TestingCycleStatus

The RV_TestingCycleStatus view provides status information for testing cycles.

You can use this view to create a *report that shows the current status of a testing cycle*.

TestingCycleID denotes the testing cycle and TesterExecutionID (as well as TesterExecutionName, UserID, CapacityInCycle, TesterLogin, TesterFirstName, TesterLastName) is used to identify the assigned tester in the testing cycle. For the tests which are not assigned to a specific tester, the UserID, CapacityInCycle, TesterLogin, TesterFirstName, and TesterLastName are null. The key of this table is the combination of the columns TesterExecutionID and BuildID.

Row	Description
TestingCycleID	Identifier of the testing cycle.
TesterExecutionID	Identifies the group of tests that are assigned to a specific tester.
TesterExecutionName	The generated name for the group of tests that are assigned to a specific tester.
UserID	The user ID of the tester.
CapacityInCycleInMinutes	The capacity for this user in this testing cycle in minutes.
TesterLogin	Login name of the tester.
TesterFirstName	First name of tester.
TesterLastName	Last name of tester.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	Identifier of the project.
BuildID	Identifier of the build.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.

Row	Description
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_ExecutionPlanStatusPerBuild

The RV_ExecutionPlanStatusPerBuild view retrieves the latest test status sums for every execution plan in context of builds.

Folders and child nodes are not considered. You can use this view to create a *report that shows the status of your test runs for each execution plan in a folder*. In contrast to *RV_ExecutionPlanStatusRollup*, this view has a slight performance advantage as no hierarchy is considered for retrieving the data. The key of this table is the combination of the columns ExecutionPlanID and BuildID.

Row	Description
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
ExecutionPlanName	Name of the execution plan.
ExecutionParentFolderID	ID of the execution plan's parent.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that the execution plan belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_EPFinishedStatusPerBuild

The RV_EPFinishedStatusPerBuild view retrieves the latest test status sums for every execution plan in context of builds. This view provides similar information as the RV_ExecutionPlanStatusPerBuild view, but only for test runs from finished execution plan runs.

Use this view to create reports that ignore currently running execution plans. Folders and child nodes are not considered. You can use this view to create a *report that shows the status of your test runs for each execution plan in a folder*. In contrast to *RV_ExecutionPlanStatusRollup*, this view has a slight performance advantage as no hierarchy is considered for retrieving the data. The key of this table is the combination of the columns ExecutionPlanID and BuildID.

Row	Description
ExecutionPlanID	Identifier of the execution plan.
BuildID	Identifier of the build.
ExecutionPlanName	Name of the execution plan.
ExecutionParentFolderID	ID of the execution plan's parent.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that the execution plan belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_ExecutionPlanStatusRollup

The RV_ExecutionPlanStatusRollup view retrieves the sums for passed, failed, and not-executed tests per execution plan or folder in context of a build.

In case of folders, the counters include the numbers from all children. You can use this view to create a *report that shows the status of all test runs in a folder*. The key of this table is the combination of the columns ExecutionFolderID and BuildID.

Row	Description
ExecutionFolderID	Identifier of the execution plan.
BuildID	Identifier of the build.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that the execution plan belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_EPFinishedStatusRollup

The RV_EPFinishedStatusRollup view retrieves the sums for passed, failed, and not-executed tests per execution plan or folder in context of a build. This view provides similar information as the RV_ExecutionPlanStatusRollup view, but only for test runs from finished execution plan runs.

Use this view to create reports that ignore currently running execution plans.

In case of folders, the counters include the numbers from all children. The key of this table is the combination of the columns ExecutionFolderID and BuildID.

Row	Description
ExecutionFolderID	Identifier of the execution plan.
BuildID	Identifier of the build.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that the execution plan belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.

Row	Description
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_ConfigurationSuiteStatus

The RV_ConfigurationSuiteStatus view lists the status counts for all configuration suites and configurations per build.

You can use this view to create a *report that shows the status of all test runs for each configuration in a configuration suite*. The key of this table is the combination of the columns ConfigurationID and BuildID.

Row	Description
ConfigurationSuiteID	Identifier of the configuration suite.
ConfigurationID	Identifier of the configuration.
ConfigurationName	Name of the configuration.
BuildID	Identifier of the build.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that this row belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.

Row	Description
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.

RV_ConfigSuiteFinishedStatus

The RV_ConfigSuiteFinishedStatus view lists the status counts for all configuration suites and configurations per build. This view provides similar information as the RV_ConfigurationSuiteStatus view, but only for test runs from finished execution plan runs.

Use this view to create reports that ignore currently running execution plans.

The key of this table is the combination of the columns ConfigurationID and BuildID.

Row	Description
ConfigurationSuiteID	Identifier of the configuration suite.
ConfigurationID	Identifier of the configuration.
ConfigurationName	Name of the configuration.
BuildID	Identifier of the build.
PassedCount	Sum of all passed tests.
FailedCount	Sum of all failed tests.
NotExecutedCount	Sum of all not-executed tests.
ProjectID	ID of the project that this row belongs to.
BuildName	Name of the build used for this test run.
BuildDescription	Description of the build.
BuildOrderNumber	Order number of the build.
BuildIsTagged	1 if the build is tagged, 0 otherwise.
VersionID	ID of the version that the build belongs to.
VersionName	Name of the version.
VersionDescription	Description of the version.
VersionOrderNumber	Order number of the version.
ProductID	ID of the product that the build belongs to.
ProductCode	Name of the product.
ProductDescription	Description of the product.
ProductOrderNumber	Order number of the product.