HP OpenView Service Quality Manager

invent

DataMart User's Guide

Edition: 1.4

for the HP-UX Operating System

April 2007

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

Contents

Preface **5**

Chapter 1 .... **7**

Overview **7**

1.1 General Description of a Data Warehouse 7
1.1.1 Terminology 7
1.2 Concepts 9
1.2.1 Data Warehouse 9
1.2.2 Using Aggregates 10
1.3 Who Should Use the Service Quality Manager DataMart? 10

Chapter 2 .... **11**

What is the Service Quality Manager DataMart? **11**

2.1 Main Behavior of the DataMart Migration Component 12
2.1.1 Periodic Reading of Logged Data – Staging Session start 12
2.1.2 On-the-fly Dimension Update 12
2.1.3 Transforming the Data 13
2.1.4 Periodic Summarization and Aggregation 15
2.1.5 Periodic Purge of the DataMart 21
2.2 Behavior of the DataMart in Particular Cases 22
2.2.1 Fact Duplication when a Measure is not Updated 22
2.2.2 Processing of the Measure Duplicates 22
2.2.3 Late Measure Messages 23
2.2.4 Data Unavailability Management 23
2.2.5 Warm Start 23
2.2.6 Support for sampling scheduling 23
2.3 Description of the Service Quality Manager Data contained by the DataMart 24
2.3.1 Dimensions 24
2.3.2 Facts 24
2.3.3 Associations Between Objects 25
2.4 Service Quality Manager Data not Contained by the DataMart 25

Chapter 3 .... **26**

Interfacing with the DataMart **26**

3.1 With SQL 26
3.1.1 Use of the DataMart Production Area Database 26
3.1.2 Sample Requests 26
3.2 With an OLAP 27

Chapter 4 .... **28**

Late calculation **28**
4.1 Objective Status Late Calculation 28
4.1.1 Overview of the OS late calculation feature 28
4.1.2 Late calculation feature reference guide 30
4.2 Performance Data Late Calculation 36
4.2.1 Overview of the performance data late calculation feature 36
4.2.2 Late calculation feature reference guide 38

Appendix A **47**

Service Health Indicators computation – SA%, MTBF, MTTR **47**
Service Availability – SA% 47
Mean Time Between Failure – MTBF 47
Mean Time To Repair – MTTR 49
Cumulated OK / Degraded / Failure Duration 49

Appendix B **50**

Production Area Database Model **50**
Static Model - Generic Data 50
Dimensions 50
Dynamic Model - Specific Data 55

Appendix C **56**

Glossary **56**

Index **60**

Preface

OpenView Service Quality Manager DataMart is part of the Hewlett-Packard solution for Service Quality Management. This manual describes how to use the DataMart in the context of the complete OpenView Service Quality Manager solution.

Intended Audience

This document is intended for those who use the Service Quality Manager DataMart, in particular D-OLAP tools designers.

A complete knowledge of the Service Quality Manager type of conveyed data is a mandatory prerequisite to fully appreciate the contents of this document. A good knowledge of Oracle and SQL is also necessary, which will allow a quick understanding of how to use the generated Oracle database.

Supported Software

The supported software referred to in this document is as follows:

Product Version	Operating System
OpenView Service Quality Manager 1.4	HP-UX 11.11

The term UNIX is used as a generic reference to the operating system, unless otherwise specified.

Typographical Conventions

`Courier` Font:

- Source code and examples of file contents.
- Commands that you enter on the screen.
- Pathnames
- Keyboard key names

Italic Text:

- Filenames, programs and parameters.
- The names of other documents referenced in this manual.

Bold Text:

- To introduce new terms and to emphasize important words.

Associated Documents

The following documents contain useful reference information:

- *OpenView Service Quality Manager DataMart Configuration and Administration Guide.*
- *TMF701, Performance Reporting Concept & Definitions V1.1.*

- *OpenView Service Quality Manager Sampling Scheduling Guide*

Support

You can visit the HP OpenView support web site at:

http://support.openview.hp.com/support.jsp

This Web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Chapter 1

Overview

1.1 General Description of a Data Warehouse

1.1.1 Terminology

This section lists the terms used in the Warehousing/Reporting area.

1.1.1.1 Warehousing

Data Warehouse

Database (usually relational) that stores a copy of operational data whose structure is optimized for query and analysis. The scope of the data warehouse is generally considered to be the entire enterprise.

DataMart

Highly focused data warehouse whose scope is usually confined to a single subject area. There is some controversy regarding the antecedents of the data mart. Some sources assert that the data mart must be derived from an enterprise-scale data warehouse, while others construct their data marts directly from operational data stores.

OLAP (Online Analytical Processing)

Business analyses based on multidimensional data models. There are different possibilities to distinguish the tools. One possibility is a database that can be relational (ROLAP tools), multidimensional (MOLAP tools) or hybrid as a combination of both (HOLAP tools). Another way is the kind of cubes being used (hypercube or multicube). Typical operations for all OLAP applications are slice and dice, pivoting and drill up/down.

Dimensional Model

Top-down design methodology that for each business process enumerates relevant dimensions and facts.

Star Schema

Special, de-normalized data model used by relational databases to provide a multidimensional structure for OLAP applications. Several dimension tables surround the fact table. With some imagination this schema looks like a star.

Dimension Table

In a Dimensional Model, table that contains data about one of the dimensions. The dimension table has a primary key that is used to connect it to the fact table. The dimension table has as many attribute fields as possible. These fields describe individual characteristics of the dimension.

Hierarchy
Organization of data into a logical tree structure. Most dimensions are ordered by a hierarchy such as the time dimension: Year - Quarter - Month - Week - Day - Hour. Elements of a hierarchy are parent members, children members and siblings.

Fact Table
In a Dimensional Model, central table that contains the individual facts being stored in the database. There are two types of fields in a fact table:

1. The fields storing the foreign keys that connect each particular fact to the appropriate value in each dimension.
2. The fields storing the individual facts - such as number, amount, or price.

The granularity of the fact table is one of the most significant design decisions in creating a data warehouse. The facts should be as detailed as possible to allow for the data to be viewed from the greatest number of perspectives.

Granularity
Level of detail of the facts stored in a data warehouse.

Aggregates
Information stored in a data warehouse in a summarized form.

Data Quality Assurance
Process to remove errors and inconsistencies from data being imported into a data warehouse.

Data Cleaning
Removing errors and inconsistencies from data being imported into a data warehouse.

Data Migration
Corresponds to the movement of data from one environment to another. This happens when data is brought from a legacy system into a data warehouse.

Data Transformation
Corresponds to the modification of data as it is moved into the data warehouse.

ETL
Stands for Extraction, Transformation and Loading.

1.1.1.2 Query/Reporting

Query and Reporting
Type of data access and analysis computer application that allows users to build queries of the database and construct reports via a GUI.

Data Mining
Process of finding hidden patterns and relationships in the data. Analyzing data involves the recognition of significant patterns. Human analysts can see patterns in small data sets. Specialized data mining tools are able to find patterns in large amounts of data. These tools are also able to analyze significant relationships that exist only when several dimensions are viewed at the same time. Data mining is needed when the user's questions are more vague or general in nature. Data mining questions would include: "What attributes characterize the customers that gave us the most business in the past year?"

Drill Down
Changing the view of the data to a greater level of detail.

Drill Up
Changing the view of the data to a higher level of aggregation.

Multidimensional Analysis, OLAP
Process of analysis that involves organizing and summarizing data in a multiple number of dimensions. People can comprehend a far greater amount of information if that information is organized into dimensions and into hierarchies. The wide use of spreadsheets and graphs illustrates the need for people to have their information organized. A spreadsheet is a two-dimensional analysis tool. If a person could comprehend 10 individual facts, they could possibly comprehend 100 facts if they were arranged in a spreadsheet. If 3 or 4 or 5 dimensions could be displayed, the amount of information that could be comprehended would be increased exponentially - to 1000 facts, 10,000 facts, and 100,000 facts. Multidimensional data is also organized hierarchically, allowing users to "drill down" for more detailed information, "drill up" to see a broader, more summarized view, and "slice and dice" to dynamically change the combinations of dimensions that are being viewed.

Business Intelligence Tools
Software that enables business users to see and use large amounts of complex data. The following three types of tools are referred to as Business Intelligence Tools:

a. Multi-Dimensional Analysis Software - Also Known As Multi Software or OLAP - Software that gives the user the opportunity to look at the data from a variety of different dimensions.

b. Query Tools - Software that allows the user to ask questions about patterns or details in the data.

c. Data Mining Tools - Software that automatically searches for significant patterns or correlations in the data.

Cube, or Multidimensional Cube
Fundamental structure for data in a multidimensional (OLAP) system. A cube contains dimensions, hierarchies, levels, and measures. Each individual point in a cube is referred to as a cell.

Cell
One individual place in a Cube.

1.2 Concepts

1.2.1 Data Warehouse

A Data Warehouse is a database where data are collected for the purpose of being analyzed.

A Data Warehouse collects, organizes, and makes data available for the purpose of analysis - to give management the ability to access and analyze information about its business. This type of data can be called "informational data". The systems used to work with informational data are referred to as OLAP.

Bill Inmon coined the term "data warehouse" in 1990. His definition is: "A (data) warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process":

- Subject-oriented - Data that give information about a particular subject instead of about a company's on-going operations.
- Integrated - Data that are gathered into the data warehouse from a variety of sources and merged into a coherent whole.

- Time-variant - All data in the data warehouse are identified with a particular time period.
- Non-volatile - Data are stable in a data warehouse. More data are added, but data are never removed. This enables management to gain a consistent picture of the business.

1.2.2 Using Aggregates

Instead of recording the date and time each time a fact happens (as a certain product is sold), the data warehouse could store the quantity of the product sold each hour, each day, or each week.

Aggregates are used for two primary reasons:

- To save storage space. Data warehouses can get very large. The use of aggregates greatly reduces the space needed to store data.
- To improve the performance of business intelligence tools. When queries run faster they take up less processing time and the users get their information back quicker.

This means that *detail* (for decision making, trend analysis, reporting) *is lost with aggregation*.

Some data warehouses store both the detailed information and aggregated information. This takes even more space, but gives users the possibility of looking at all the details while still having good query performance when looking at summaries.

Some systems use aggregates for historical data. Perhaps detailed data is kept on-line for a year. After that the detailed data is kept in a less accessible, permanent storage format, and only the aggregated, summary data is kept on-line.

There are five aggregate functions defined in standard SQL (Structured Query Language): SUM, COUNT, MIN (the lowest value), MAX (the highest value), and AVG (the average value).

1.3 Who Should Use the Service Quality Manager DataMart?

The Service Quality Manager DataMart is intended for people who want to develop end-user applications on top of the Service Quality Manager statistical data, that is, mainly the service management performance data and quality of service data.

Typical users of the Service Quality Manager DataMart are end user applications development managers, programmers and business analysts for the data warehouse.

Although direct access to the data is possible by using for example the SQL interface (see Chapter 3), the major use is expected through OLAP software such as Business Objects, Query Tools, or Data Mining Tools. Section 3.2 provides an example of how an OLAP can be integrated on top of the Service Quality Manager DataMart.

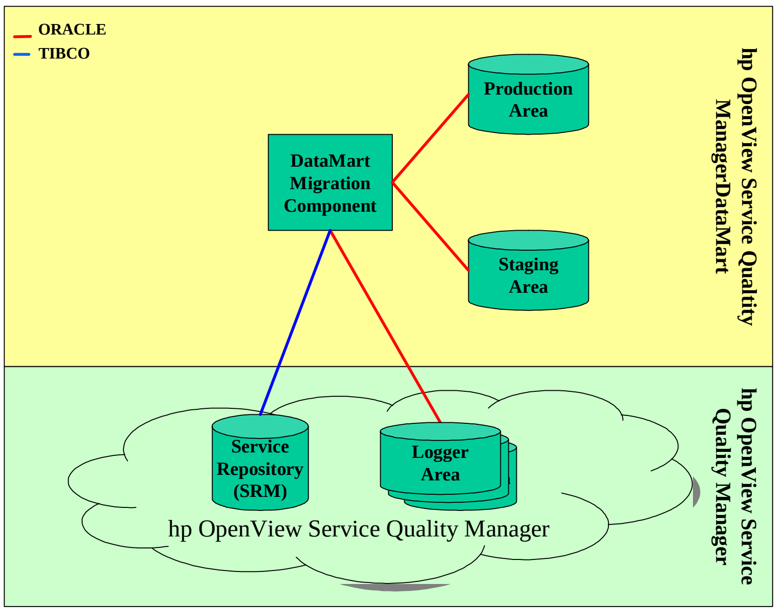
Chapter 2

What is the Service Quality Manager DataMart?

The Service Quality Manager DataMart provides the facts and the dimensions containing a majority of the data handled within Service Quality Manager.

The Service Quality Manager DataMart architecture can be shown as follows:

Figure 1 General Architectural Overview


ORACLE
TIBCO
Production Area
DataMart Migration Component
Staging Area
hp OpenView Service Quality ManagerDataMart
Service Repository (SRM)
Logger Area
hp OpenView Service Quality Manager
hp OpenView Service Quality Manager

The Service Quality Manager DataMart is composed of the following components:

- The ***DataMart Migration Component***: this is a software program that performs the whole processing including the logger area records reading and processing, and the production area construction.
- The Oracle databases: the *production* and the *staging* areas are Oracle databases created and filled by the Service Quality Manager DataMart Migration Component.

  The *production area* is the main database that contains all the produced data and is the one to be used by any reporting functions.

  The *staging area* is a working database, which is of no interest for the Service Quality Manager DataMart end user.

This chapter describes the behavior of the DataMart Migration Component, with a distinction made between the main processing and the particular cases. Then the Service Quality Manager data that are contained and those that are not contained in the data mart are briefly listed.

2.1 Main Behavior of the DataMart Migration Component

The Service Quality Manager DataMart functions as follows.

When started, the DataMart Migration Component reads its configuration data: database names and passwords, scheduling parameters and so on. For details concerning the configuration, refer to the *OpenView Service Quality Manager DataMart Configuration and Administration Guide.* This document also describes how to update the configuration.

2.1.1 Periodic Reading of Logged Data – Staging Session start

Once started, the DataMart Migration Component launches a timer that wakes it up in order to regularly (typically, every 15 minutes) read the new Service Quality Manager data made available in a dedicated database: the Logger Area(s). The read information is then transformed into raw facts that are finally stored into the Production Area.

Note

The logged data are always updates compared with the previous state of the service management data (model, performance data, objective statuses).

If the DataMart Migration Component is plugged on an already running OV SQM environment, it will only reflect a full consistent view of the data once all the data have been updated at least once.

Therefore, it can take several staging sessions before the Production Area contains a full description of operational service management data.

2.1.2 On-the-fly Dimension Update

The dimensions are updated "on-the-fly", that is, each time a yet-unknown object (for the DataMart) is discovered or dealt with, it is added into the corresponding dimension. For example, when a message concerns a customer, the DataMart Migration Component searches for this customer in the Customer dimension. If the customer is not found, it is automatically added after having requested the customer characteristics (label, description and so on) to another Service Quality Manager module.

With the dynamic dimensions (Service Definition specific tables), the same processing is used, that is, an on-the-fly update. They are created and updated on-the-fly as well.

According to the ‘Init Model At Startup’ flag, described in *Datamart Installation, Configuration and Administration Guide* the model can be loaded at DataMart startup. This limits runtime discovery and provides full dimensional information to reporting tools even if no collection has impacted the model entities.

2.1.2.1 When an Object is Deleted

When an object that is contained in a dimension is deleted from the Service Management model, the DataMart Migration Component is informed by an update message. This object is then marked as deleted in the dimension by setting the column

"IS_MARKED_AS_DELETED" to 1 (True) and setting the "DELETION_TIMESTAMP". This object is then seen as deleted for the DataMart Migration Component.

Note

The deletion is not effective in the database since reports, especially on summarized data, may still be accessed on formerly managed objects.

That is why the only action is to flag the object as being deleted.

2.1.3 Transforming the Data

The Data Transformation corresponds to the modification of data as they are moved into the actual data warehouse – also named the 'real' DataMart or Production Area database. Generally, data transformation in a warehouse can include:

- Data cleaning – this is part of the process of data quality assurance.
- Normalization – organizing the data into the normal structure of a relational database (dimensions and facts).
- Processing calculations.
- Changing data types.
- Making the data more readable.
- Replacing codes with actual values.
- Summarizing the data by various time periods - See Section 2.1.4.

Concerning the DataMart Migration Component, besides summarization (which is dealt with below), the main transformations it performs are:

- The time identifier determination from a message timestamp.
- The object identifiers recovering from their name.
- The enumeration management.

All this is performed in order to reduce the fact data storage space and accelerate reporting access to the facts, which are key requirements in Data Warehousing.

2.1.3.1 Determining the Time Identifier

By definition, Data Warehousing facts are time stamped. The Service Quality Manager DataMart timestamps correspond to either of:

- Data collection timestamp: performance data.
- Calculation timestamp: objective statuses.
- Event emission timestamp: an SLA is updated.

These timestamps are to be determined as belonging to a time interval in order to group the messages for reporting purposes. The DataMart Migration Component works with the general time dimension which contains one record per time interval, that is, at each **DataMart Granularity**. For example, if the DataMart Granularity is 5 minutes, the time dimension contains one record for the interval 0 (included) to 5 (excluded) minutes, another for 5 to 10 minutes, and so on.

Time stamping the messages therefore consists of:

- Extracting the timestamp from the message.
- Determining the corresponding time interval depending on the DataMart Granularity.

- Searching the associated 'time_id' from the time dimension table.

The records inserted into the fact tables are time stamped this way.

When several messages for the same entity are in the same time interval

In this case, the DataMart does not elect any of the messages. It simply stores all the received messages. It is the responsibility of the reporting function that is plugged on top of the production area to choose among the generated facts those that are of interest.

Note about the Timestamp Time Zone

At the DataMart Level al the timestamps are expressed in **GMT**, the conversion to local time might be done by the GUI or the reporting function. However local time is used by the DataMart to define the aggregation periods like days, weeks…

2.1.3.2 Recovering Dimension Identifiers

All the dimensions containing the definition of the Service Quality Manager objects – Service Instance, Service Definition, SLA, and so on– contain the NAME-s of the objects they store. When the name is not self-discriminatory (for example, in case of a Service Component Definition name that is unique only within a certain Service Definition), the dimension has a column called 'global name' that contains a concatenation of the necessary container names; in the case of an SCD, the global name column would be equal to '<SDName>.<SCDName>'.

These dimensions associate an identifier to the data describing the objects (label, description and so on). These identifiers are used to make reference (from the fact tables) to the objects, hence improving the access performance from the fact tables: it is faster to make a search with a numeric value than with a string.

While reading the logged data, if necessary, the global name of the relevant object is rebuilt and is used to recover the correct identifier from the dimension. This identifier is then used to identify the object from the fact table.

2.1.3.3 Enumeration Management

A single enumeration labels dimension table is used to store all the used enumeration values. This avoids repeating the enumeration labels in the fact tables (saving disk space, accelerating data access).

When processing a yet unknown enumeration value therefore, the DataMart Migration Component adds the associated entry in the enumeration labels dimension.

2.1.3.4 Number Management

Datatypes like Int, Float and relative times are can be managed in two ways.

2.1.3.5 Adopted Conversions

The following table details how the Service Quality Manager DataMart stores the parameters and properties in its Oracle databases.

Table 1 Adopted Conversions

XML Datatype	Oracle Datatype	Comments
INT	Number(38)	Maximum size for a number with Oracle (38 digits length).
FLOAT	Number	

XML Datatype	Oracle Datatype	Comments
ENUM	Number(4)	Possible values: 0 to 9999.
STRING	Varchar2(n)	
DATETIME (ABSTIME)	Date	Default format: 'DD-MON-YY HH:MM:SS' . This example default date format includes a two-digit number for the day of the month, an abbreviation of the month name and the last two digits of the year.
DATETIME (RELATIVE TIME)	Number(38)	The relative time data are stored as integers in Service Quality Manager DataMart.

2.1.4 Periodic Summarization and Aggregation

2.1.4.1 Summarized Time Dimensions

In order to refer to the information stored in the time dimension, the time identifiers of the beginning of the relevant summarization period are kept in the summarized fact tables. This allows quicker access time from the reporting tool. Sometimes this allows simpler report development with OLAPs such as BusinessObjects © (AggregateAware™ function).

For example:

- For the hourly summarized fact tables, the time identifier of the record corresponding to the beginning of the hour (hh:00) is kept.
- For the daily summarized fact tables, the time identifier of the record of the day at 00:00 is kept.
- For the monthly summarized fact tables, the time identifier of the record of the first day of the month at 00:00 is kept.
- The processing is alike for all the other summarization periods (weekly, quarterly, yearly) fact tables.

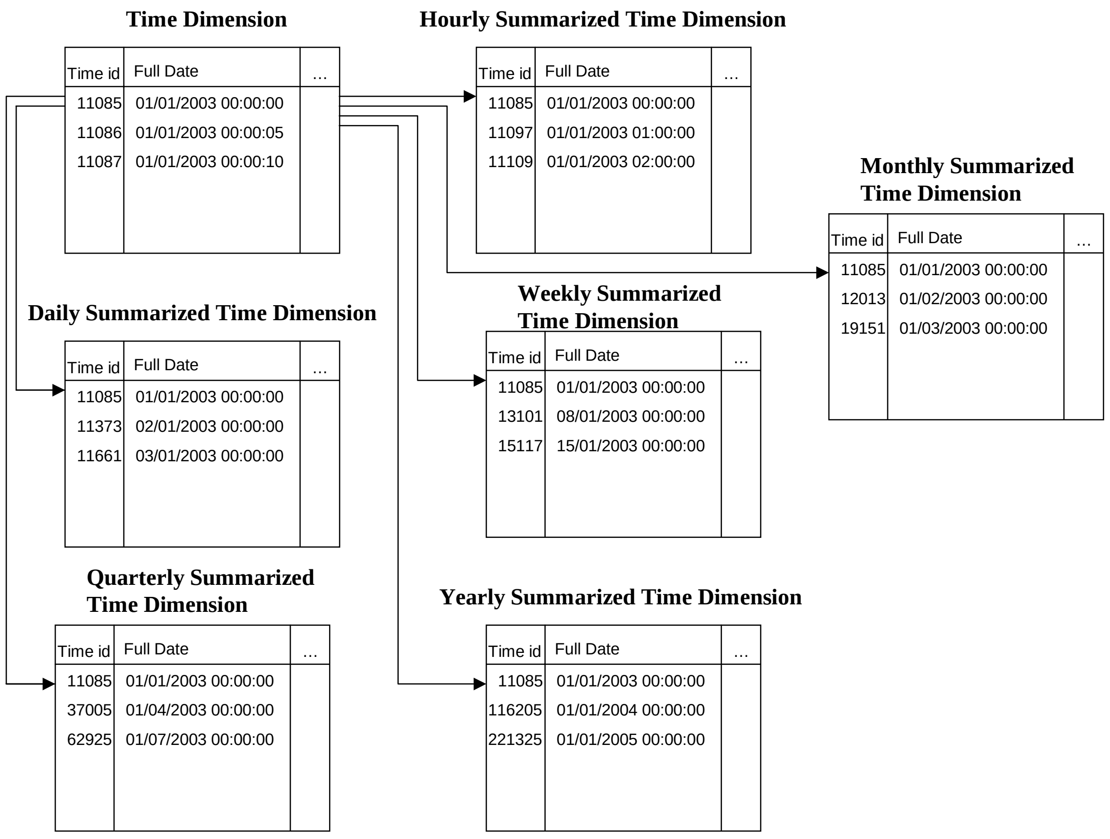
For this, one summarized time dimension is created per summarization period. The resulting tables contain only useful information relating to the summarized period, that is:

- The hourly summarized time dimension contains one record per hour.
- The daily one contains one record per day.
- …

In order to allow the restoration and to link the summarized data, it is useful to link the summarized time dimensions together. For this purpose, the summarized time dimensions Time_ids use Time_ids of the "normal" time dimension (Time_dimension).

The following figure can help to understand how the summarized time dimensions are linked with the general ("normal") time dimension.

Figure 2 Time Dimension and Summarized Time Dimensions



2.1.4.2 Performance Data Summarization

The data summarization algorithms cannot be configured.

Numeric Values [1]

For a set of identified fact tables, each time a new fact is summarized; the DataMart Migration Component calculates and stores a set of summarization fields. These values are the minimum value, the maximum value and the average value or the sum of values obtained for the current period (hour, day, month, year depending on the fact tables).

Therefore, the Service Quality Manager DataMart stores both the *detailed* information (*raw data*, for each DataMart Granularity) and the *summarized* information (for different periods).

The way the DataMart summarizes numeric values depend from their semantic and is driven by the parameter's category in the model.

The datamart provides either a Sum aggregation for parameters with a 'counter' category that are considered as events. This means that when no new value is received for a time granularity nothing is added to the Sum. Only received measures are aggregated. The others categories of parameters behave as states, so a parameter is considered to keep its value until a new one is received. The Datamart provides minimum, maximum and average values weighted by the duration.

[1] Numeric means in Service Quality Manager DataMart: integer or float values.

The performance data are summarized per hour, day, month, quarter and year. The summarized data are kept in different summarized fact tables (one per period) and can then be accessed directly by the Reporting functions (OLAP, SQL and so on).

In parallel to the fact table updates, the summarization tables for performance data (Parameters) have to be updated. Their structure is the same as the fact tables except that, for each parameter, there are 3 columns: minimum value, maximum value and average value or 1 Sum column value.

Therefore, each time a fact table structure is updated (addition of a column); the corresponding summarization table has to be updated (addition of three columns). The calculations executed are detailed in the following sections.

Minimum Value

Each time a new record is summarized in a fact table, each added numeric value is compared to the current minimum value (of the corresponding column) of the summarized table, for the summarized period. When the new value is lower than the stored minimum value, it is taken as the new summarized minimum value. It is then stored in the summarized fact table.

Maximum Value

Each time a new record is summarized in a fact table, each added numeric value is compared to the current maximum value (of the corresponding column) of the summarized table, for the summarized period. When the new value is higher than the stored minimum value, it is taken as the new summarized maximum value. It is then stored in the summarized fact table.

Average Value

Each time a new record is summarized in a fact table, the collection duration for each summarized period is increased. Each added numeric value is then integrated in the following formula, updating the average value of each column. The average formula is common:

New Average = ((Current duration) * (Current average) + (New value)) / (New duration)

This calculated value is stored as the new average value.

Note

The duration used to calculate the average does not include the periods where no data is available for the parameter.

Unavailability Duration Value

The time elapsed while no data is available for a parameter (indicated as noValue=”True’ in the XML payload) is computed and stored in this indicator.

Sum Value

Each time a new record is summarized in a fact table it is added to
the already totalized facts in the summarized tables

Enumerated Values

Processing a new enumeration value

When an enumeration parameter value is being added to a fact table, that is, the parameter has this value for the fact being processed, the corresponding summarized column is added. For example, if a parameter value is "Enabled" (for an attribute called "Operational State"), the column "NUMOF_ENABLED" is added to the summarized tables.

Summarizing the enumeration values

Depending on the value of the parameter datatype that is an enumeration, the DataMart Migration Component increments the corresponding column. The summarized fact tables are therefore filled with the number of each value counted during the summarization period. For example, when a logged message contains the collection operational state equal to "Enabled", the column "NUMOF_ENABLED" is incremented.

This way, the number of occurrences of each state (or enumeration value) is directly available and it is then easy to build up statistics.

Note

The number of occurrences is kept instead of a percentage computed on-the-fly to avoid any loss of precision.

Warning

A limitation has to be taken into account: ORACLE allows the user to have at most 1000 columns in a table. The DataMart Migration Component does therefore not process more than a configurable number of different values. See the configuration field "Enumeration Maximum Number of Values" for information in the *OpenView Service Quality Manager DataMart Configuration and Administration Guide*.

When exceeded, the DataMart Migration Component does not summarize the new enumeration values anymore.

Absolute Time

No summarization is offered for the absolute time. Only the raw performance fact tables (non summarized data) provide the values for such parameters.

Relative Time

The relative time represents a number of seconds and is stored as an integer. For this reason, the same rules are applied as for the integer to the relative time, see Section 2.1.3.5.

2.1.4.3 Service Health Indicators Aggregation – SA%, MTTR, MTBF…

One of the main features of Service Quality Manager is its ability to compute the well-known Service oriented indicators such as the Service Availability percentage (SA%), the Mean Time Between Failure (MTBF), the Mean Time To Repair (MTTR). These indicators are called *Service Health Indicators* or SHI.

At each Staging Session, the Service Health Indicators are computed for each aggregation (also called summarization) granularity: daily, monthly, quarterly and yearly. This way, the data are always up to date; for example it is not necessary to wait for the end of a month before having the data available for the quarter.

This section explains these indicators. The standard formulas are adapted to the specificity of Service Quality Manager. Appendix A details the formulas and shows how they are obtained.

Note

The service health indicators aggregation algorithms cannot be configured.

Service Degradation Factor and Objective Status

As a reminder of Service Quality Manager general concepts, the Service Degradation Factor represents the degradation of a service.

It can vary from 0 (operational) to 1 (failure), the intermediate values indicating in which proportion the service is altered (degraded).

The Objective Status value is equal to:

```
1 - Service Degradation Factor
```

The Service Degradation Factor is used for the definition of the Service Level Objective / Threshold; otherwise, Service Quality Manager makes use of the Objective Status. That is, it provides the degradation information at a given time, within a certain Service Level Objective and Agreement, of a Parameter, a Component, a Service, or an SLA.

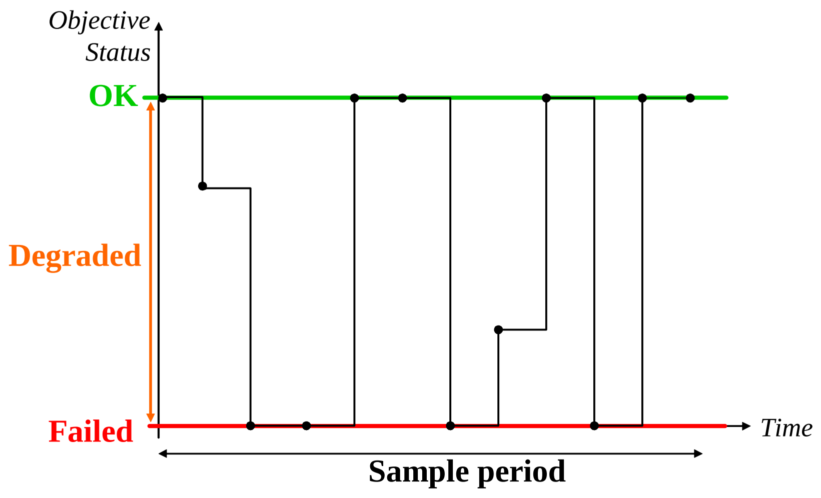
The Objective Status is the reflection of the worst Service Degradation Factor of the hierarchy (modulo the associated weight).

Note

The service health indicators are computed taking into account the "In Service Hours".

For example, if the data collection interval is 5 minutes, and the reference period is the last hour, the following chart can be obtained:

Figure 3 Objective Status Evolution Chart



Service Availability Percentage – SA%

The Service Availability Percentage is an aggregation of various individual objective statuses for a specific period of time (for example, on a monthly basis). It reflects the provided quality of service. The Service Availability is expressed as a percentage (SA%) indicating how the QoS objectives were met for a given period, on average.

The following data are used to calculate the Service Availability percentage within a given Service Level Objective for a Parameter, an SCI/SI, a CAView, or an SLA, for a given period:

- The successive Objective Status values – OS – through the given period.
- The duration of each period - *Delta T.*

The following formula is used:

$$SA = \frac{\sum(OS * \Delta T)}{\sum(\Delta T)}$$

Refer to Appendix A for further information on this formula, including examples of the generated Service Health Indicators table.

Mean Time Between Failure – MTBF

The MTBF is an indicator of the delivered service quality reliability on a statistical basis from the known violations of Service Level Objectives on the service and its components. In Service Quality Manager, these components are (parameters, SIs, SCIs, SLAs). The calculation is given by the following formula. The result is expressed in hours.

$$MTBF = \frac{\Delta T}{number(FailedPeriods)}$$

Delta T is the duration during which the system is monitored since the beginning of the summarization period (day, month…). This value is therefore reset to 0 at the beginning of each summarization period.

A *failed period* is the duration of a set of consecutive facts when the Objective Status of the object (parameter, SI, SCI, CAView, SLA) is lower than or equal to the Violation State Level.

Refer to Appendix A for further information on this formula, including examples of the generated Service Health Indicators table.

Mean Time To Repair – MTTR

The MTTR is the average time between a failure and the completion of repair[2] in a large population of identical systems, components, or devices. In Service Quality Manager, these components are (parameters, SIs, SCIs, SLAs).

The MTTR corresponds to the total corrective maintenance time divided by the total number of corrective maintenance actions during a given period of time. The result is expressed in hours per corrective maintenance action.

$$MTTR = \frac{\sum(FailureDuration)}{number(FailedPeriods)}$$

Refer to Appendix A for further information on this formula, including examples of the generated Service Health Indicators table.

Cumulated OK/Degradation/Failure Duration

The computation of these three indicators is straight forward: starting from the first Time Id of a considered aggregation period (month/quarter/year…), each fact with a particular state implies the addition of one DataMart Granularity to the corresponding accumulation column.

[2] A failure is a violation, and a completion of repair corresponds to restoring the delivered QoS to the QoS that is committed to in the SLA.

2.1.4.4 Service Compliance Indicators Aggregation

Service Compliance indicators are computed as Service Health indicators and also take into account in Service hours. All the above described indicators are provided as compliance indicators. The only difference is that aggregation is performed over the Service Level Agrement reference period. For instance, with a quarterly reference period, failure duration will be accumulated over the quarter and not the data granularity as for SHI indicators. Compliance indicator can therefore show compliance evolution over the reference period at various granularities.

Monthly cumulated failure can be monitored over the quarter...

The notions of reference period and maximum violation duration of an SLA also define additional indicators that can be deduced from usual compliance indicators.

These additional compliance indicators like compliance percentage or time left before compliance violation aren't provided as such by the DataMart but are computed by the reporting according to the SLA properties and the usual provided Compliance indicators aggregated over the SLA reference period. Refer to the *Reporting users guide* for a detailed description of compliance reports

2.1.5 Periodic Purge of the DataMart

The purge function of the DataMart Migration Component processes the *Production Area* database.

Every day at 01:00 hour, and depending on the *Production Area Management Mode* (Delete/NoAction) the Service Quality Manager Production Area is processed.

The facts that are affected by the purge are those for which the creation timestamp (that is, the timeid) is older than the associated storage duration configuration field specified in the DataMart Migration Component configuration and is not needed to calculate unterminated aggregation period.

The action is applied to all the *fact* tables. The age is specific to each type of fact table: raw data, hourly (daily…) summarized data.

In the case of a *delete* action, the Production Area expired fact data are simply deleted from the Production Area database.

The dimension tables aren't affected by the purge, dimensional information is always kept to be able to report on facts even for deleted dimensions.

A fact is considered as needed to calculate unterminated aggregation period if no other fact has been received and summarized in the succeding period. For exemple, if the last fact has been received at 14:20 it is needed untill a fact is received after 15:00 because it closes the 14:00 to 15:00 period.

The same principle is used for aggregated data:

- An hour is not purged if its enclosing day is not terminated.
- A day is not purged if its enclosing month or week is not terminated.
- A month is not purged if its enclosing quarter is not terminated.
- A quarter is not purged if its enclosing year is not terminated.

2.2 Behavior of the DataMart in Particular Cases

2.2.1 Fact Duplication when a Measure is not Updated

At every Service Quality Manager DataMart Collection Interval, only *updates* of the measures (performance data, objective statuses…) are provided to the DataMart Migration Component. However, in order to allow for easy browsing and processing of the raw **performance** and **objective status** data, the reporting tools require continuous representation of the data: **one fact at each DataMart Granularity**. Indeed, as soon as an object (SLA, Service, Component…) is given a measure (Objective Status, parameter values), it is sometimes very difficult for a reporting tool to manage if the object does not have a measure at each granularity.

For example, if the SLA "Gold VDO Customer1" has an Objective Status equal to 90% at time "01:05", and if that value is updated only at "02:35", and so on, the reporting tool will have difficulty representing the variation.

Computation of summarized or aggregated data is also eased with continuous measures.

Therefore, such measures have to be duplicated for each DataMart Granularity by the DataMart Migration Component when no updates have been notified between two staging sessions. This allows it to be shown that these measures have not changed. For consistency reasons, when certain conditions happen, these measures *are not duplicated anymore*. These conditions always imply that updates on the measures will not be received anymore:

- An object (SLA, Customer, Service, Component) referenced in the measure was deleted since the last staging session.
- An object is updated extensively enough, implying that the measure is not a candidate to be duplicated anymore:

  A parameter is removed from a service definition: the performance data table column is not deleted (in order to not remove the previous values), but the value of the corresponding column is set to NULL.

  The SLA changes of a Customer, a Service or a Component are removed from the definition of the SLA: the objective status facts for the old customer/service/component of the SLA will not be duplicated anymore.

Note

Typically, during a complete or partial restart of the Service Quality Manager, the DataMart Migration Component is restarted last, after all the other components, so as to not duplicate too many records. Likewise, and for the same reason, it is stopped first, before the other components.

2.2.2 Processing of the Measure Duplicates

In Service Quality Manager, due to the TIBCO architecture that cannot ensure a message is not emitted twice, the same measures can be logged more than once (duplicated) by the system, and hence be processed more than once by the DataMart Migration Component.

A duplicated message is processed as any other message. It is a malfunction of the global system that has to be seen as exceptional.

Therefore no special processing is foreseen in such cases.

The Service Quality Manager TIBCO architecture does not prevent receiving duplicate messages. Duplicate messages may alter the real data (such as a parameter value average) if they are processed normally, but this side effect is admitted within Service Quality Manager. Moreover, this side effect is not significant from a statistical point of view.

2.2.3 Late Measure Messages

Since SQM v1.3, the DataMart handles desynchronized parameter within the same component definition and keeps the exact timestamps. When a new value is received for this parameter this assumption is natively and automatically corrected.

2.2.4 Data Unavailability Management

Before SQM 1.4, the datamart used to ignore data unavailability notifications (noValue="True" in the XML payload) assuming that the performance parameter was keeping its previous value. In SQM 1.4, those messages can be stored and processed, and a new indicator is now proposed in summarized tables showing the unavailability duration of each summarized parameter.

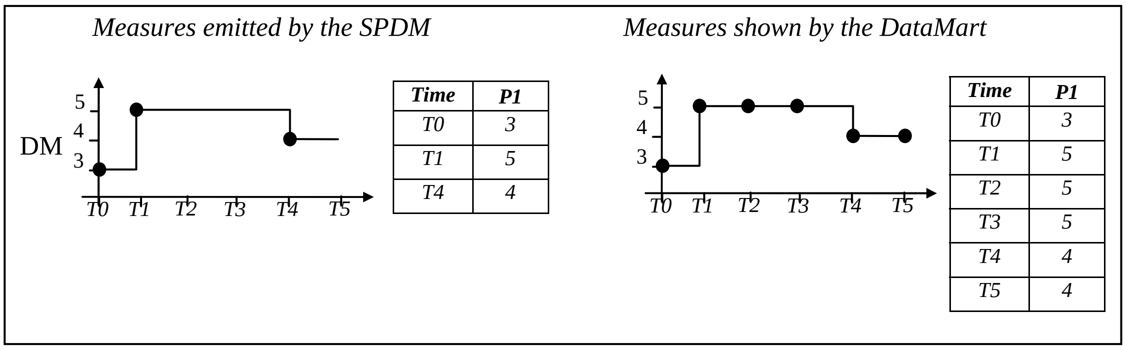
2.2.5 Warm Start

When restarted, the DataMart Migration Component simply schedules itself as done after a cold start, that is, prepares for the periodic reading of the logged data. Also, as for its first start, the DataMart Migration Component does not retrieve the data for the objects that may have been created before being restarted.

Until a new update message is processed by the Service Quality Manager DataMart, the previous values are simply duplicated. See Sections 2.1.1 and 2.2.1.

2.2.6 Support for sampling scheduling

The DataMart supports the sampling scheduling feature. In consequence, if no measure is received at sample timestamp for a given parameter the DataMart "duplicates" the measure of the preceding sample timestamp.



Time	*P1*
T0	*3*
T1	*5*
T4	*4*

Time	*P1*
T0	*3*
T1	*5*
T2	*5*
T3	*5*
T4	*4*
T5	*4*

2.3 Description of the Service Quality Manager Data contained by the DataMart

This section briefly describes the Service Quality Manager data the Service Quality Manager DataMart stores in the Production Area database. A complete description of all the generated data the Production Area contains is given in Appendix B.

The Service Quality Manager DataMart stores the following data in the Production Area database:

2.3.1 Dimensions

The dimensions are table data which do not evolve often, if at all. Several types of dimensions are present in the Service Quality Manager DataMart Production Area database:

- Objects of the Service Management Object Model: Service Definition, Service Component Definition, Parameter, Service Instance, Service Component Instance, Customer, Service Level Agreement, Service Instance Group, Service Level Objective, Service Parameter Objective, Objective Threshold.
  Each of the associated dimension tables contains the following data fields: Identifier, Name, Label, Description, Global Name (when required to uniquely identify an object, for example a Parameter), and specific fields that define the object, for example, the type for a Customer.
- Enumeration Datatype values: as seen in Section 2.1.3.3, there is a single table containing all the enumeration values and associated labels for all the enumeration datatypes of the parameters.

2.3.2 Facts

Each fact table provides common fields that can be grouped as follows:

- ID: the Primary Key. It uniquely identifies a fact within its table.
- TIME_REF: points to the associated (normal or summarized) time dimension. This represents the "aggregated" time at which the fact occurred.
- Dimension Identifiers: these are foreign keys linking the fact to the associated objects.

There are two types of facts:

- Raw facts: usually such types of facts are intended to be stored for a short time (typically, a month or even less). Model updates events, status updates events, performance values, objective status values, and crossed parameter events are the distinct raw data facts the DataMart Migration Component stores.
- Summarized and/or Aggregated data: in order to reduce the necessary storage and accelerate the reporting functions, the raw fact data are summarized/aggregated. Refer to Section 2.1.4 for more information.

Inventory History

The aim of the Service Quality Manager DataMart specifically and of a Data Mart in general is not to provide a full historical inventory database. Other types of software are dedicated to such requirements. The Service Quality Manager DataMart essentially contains statistical data, see Section 2.3.3.

Also refer to Appendix B for a complete description of the Oracle tables the DataMart Migration Component fills – stored in the Production Area database.

2.3.3 Associations Between Objects

Some associations between objects can evolve: for example, the Service "Video Paris main" of a particular SLA is replaced by "Video Paris backup". According to the aim of the reporting function which processes statistics, hence data that usually concern a period of time rather than an instant, it is not meaningful to keep such information as: "currently the SLA sla1 is composed of Services s1 and s2". Such non-permanent associations could still be stored in the dimensions, but as just mentioned, this is not useful or practical for statistics reporting purposes.

However, it is still definitely meaningful to provide statistics that are valid during the period of time when these objects are associated, that is, those that give the context under which they were computed.

For example, the Service Quality Manager DataMart aggregated fact tables provide the Service Health Indicators concerning the associations between objects existing at the time the measures and computations were done. For instance, the SLA level Service Health Indicators aggregated tables provide the indicators computed while an SLA and a Customer are associated. If the Customer of an SLA is changed during a summarization period, the indicator values concerning the SLA and its former Customer will not evolve anymore – although still stored, and contrary to that, the indicators concerning the SLA and its new Customer will then be computed instead.

This results in the following logic: **the facts contain the information on changing associations between objects, the dimensions do not**.

Caution

Few exceptions are like the following: the reference to the current Customer of an SLA is stored in the SLA dimension.

Note

Permanent associations, that usually represent the core definition of the objects, are stored in dimensions. For instance, the Service Definition of a Service is part of the Service dimension.

2.4 Service Quality Manager Data not Contained by the DataMart

All the Service Quality Manager data contained by the DataMart are listed in Appendix B.

The following types of data are not present in the Production Area Database:

- Data Feeder Definition, Instance, Bindings
- Subscriber Naming Service
- Calculation Expressions
- SMS Action Executors

Chapter 3

Interfacing with the DataMart

The typical use of the Service Quality Manager DataMart is with an OLAP tool such as BusinessObjects.

However, since the Service Quality Manager DataMart produces and builds an Oracle database, it is still possible to retrieve the data with the use of SQL statements.

This chapter describes these two ways of accessing the data:

- With SQL
- With an OLAP, such as BusinessObjects

3.1 With SQL

3.1.1 Use of the DataMart Production Area Database

In order to write SQL statements, you first need to understand the Production Area Database contents. Refer to Appendix B for a full description of this database.

A good knowledge of the SQL programming language is also necessary.

3.1.2 Sample Requests

These are SQL sample requests to extract data from the Production Area Database, first on the generic data tables, and second on the Video Service Sample Model specific tables.

3.1.2.1 On the Generic Data

The following request lists the number of customer degradations and violation events for the month of February 2003:

```
SELECT
  CUSTOMER_DIM.NAME,
  Sum(CROSSED_PARAM_CUSTOMER_MONTH.NUMBER_OF_DEGRADATION_STARTS),
  Sum(CROSSED_PARAM_CUSTOMER_MONTH.NUMBER_OF_VIOLATION_STARTS)
FROM
  CUSTOMER_DIM,
  CROSSED_PARAM_CUSTOMER_MONTH,
  MONTHLY_TIME_DIM
WHERE
  ( CUSTOMER_DIM.ID=CROSSED_PARAM_CUSTOMER_MONTH.CUSTOMER_REF )
  AND  (
MONTHLY_TIME_DIM.ID=CROSSED_PARAM_CUSTOMER_MONTH.MONTHLY_TIME_REF
)
  AND  (
  ( to_char((  MONTHLY_TIME_DIM.FULL_DATE ), 'MM/YYYY')
 = '02/2003'  )  )
GROUP BY
  CUSTOMER_DIM.NAME
```

3.1.2.2 On the Video Service Sample Model

The following request gives the daily number of downloaded movies evolution for Customer 'hp’, and the Video service 'Paris’, for the month of February 2003.

```
SELECT
  DAILY_TIME_DIM.FULL_DATE,
  DAILY_TIME_DIM.DAY_NUMBER_OVERALL,
  SCI_VIDEO_D.NBDWNDMOVIES_MIN,
  SCI_VIDEO_D.NBDWNDMOVIES_AVG,
  SCI_VIDEO_D.NBDWNDMOVIES_MAX
FROM
  DAILY_TIME_DIM,
  SCI_VIDEO_D,
  CUSTOMER_DIM,
  SCI_VIDEO_DIM
WHERE
  ( SCI_VIDEO_D.CUSTOMER_REF=CUSTOMER_DIM.ID  )
  AND  ( SCI_VIDEO_D.SI_OR_SCI_REF=SCI_VIDEO_DIM.ID  )
  AND  ( SCI_VIDEO_D.DAILY_TIME_REF=DAILY_TIME_DIM.ID  )
  AND  (
  ( to_char((  DAILY_TIME_DIM.FULL_DATE ), 'IW/YYYY')
 = '02/2003'  )
  AND  ( ( CUSTOMER_DIM.NAME ) = 'hp'  )
  AND  ( ( SCI_VIDEO_DIM.NAME ) = 'Paris' )
  )
```

3.2 With an OLAP

As an example, we refer to the Service Quality Manager Reporting function. This function shows how the Service Quality Manager DataMart Production Area Database is used to build graphical (HTML, PDF…) Service Quality Management reports that can be accessed with the use of a Web browser, or even published by email.

Chapter 4

Late calculation

The late calculation feature provides the ability for the DataMart to replace data, coming from the normal SQM collection and computation, by raw data provided by user. The aggregated tables are then updated. This feature applies to Objective Status data and Performance Data.

4.1 Objective Status Late Calculation

4.1.1 Overview of the OS late calculation feature

The DataMart OS late calculation feature gives the ability to modify aggregation results at Service Level Agreement, Service Component Instance and Service Parameter levels by integrating precalculated corrective events to the existing DataMart data.

4.1.1.1 Architectural overview

In order to avoid concurrent access problems which could disturb the DataMart, the late calculation processing has been delegated to the DataMart itself during the summarization process. The command line tool places the user's corrections in a queue to be processed by the DataMart at each summarization.

Step 1: Load the corrective Objective Statuses

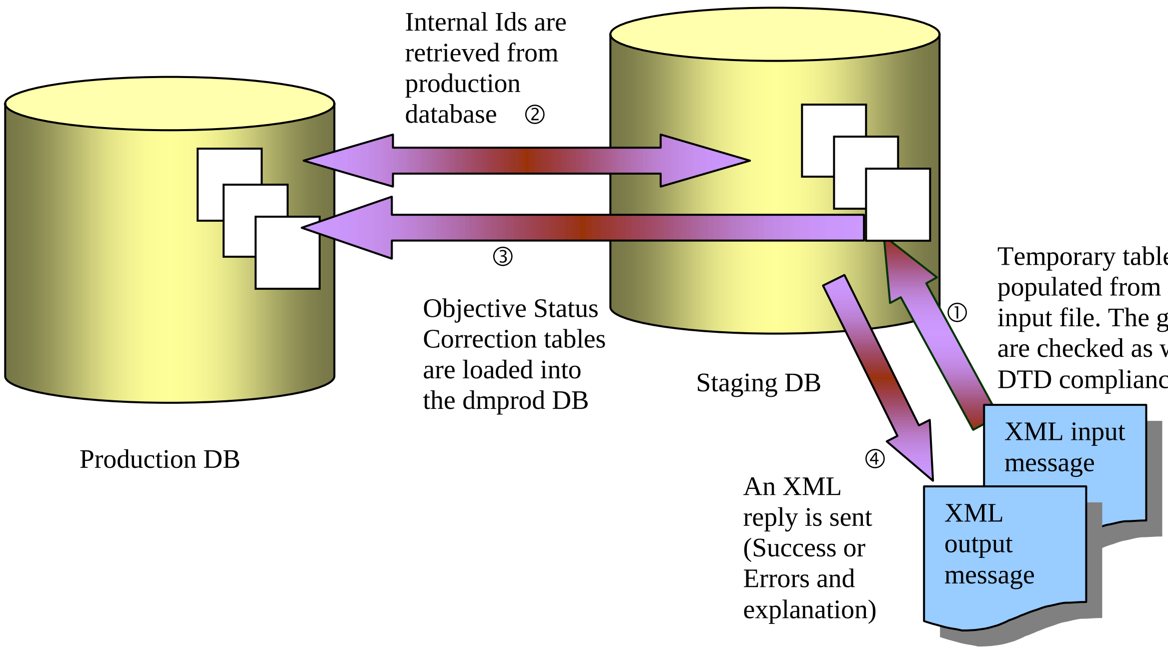
The corrective Objective Statuses are stored in an XML file to be built by the user and loaded with the *sqm_dm_load_os_correction* command line tool. The XML file must be compliant with the DTD provided by Service Quality Manager. There are two possible message types:

- message 92, used to load objective status corrections
- message 93, used to load Service Level Agreement InDuty status corrections.

The command line tool will:

- Parse and validate the XML input file.
- Populate temporary tables in staging
- Retrieve Internal IDs from the production database
- Check the dates validity
- Load correction tables into production database

The following schema shows the correction loading process:

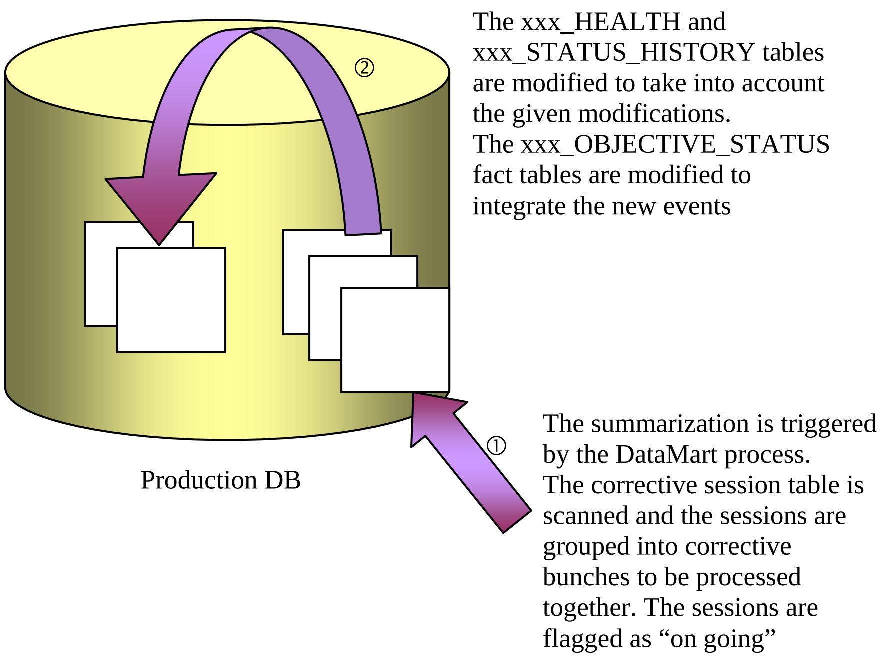


Internal Ids are retrieved from production database ②
③
Objective Status Correction tables are loaded into the dmprod DB
Staging DB
Temporary tables are populated from the given input file. The given dates are checked as well as the DTD compliance
①
XML input message
XML output message
④
An XML reply is sent (Success or Errors and explanation)
Production DB

Step 2: Process the corrections

Once loaded and validated into the production database the DataMart will process the corrections during its summarization process. This will be done at the end of the process after the normal Objective Status processing. Each XML input file has led to the creation of what the DataMart calls a corrective session.

At summarization time the datamart groups the corrective sessions involving different keys SLA/SI/SCI/SLO/Param in corrective bunches together and processes them at once.



The xxx_HEALTH and xxx_STATUS_HISTORY tables are modified to take into account the given modifications. The xxx_OBJECTIVE_STATUS fact tables are modified to integrate the new events
②
①
The summarization is triggered by the DataMart process. The corrective session table is scanned and the sessions are grouped into corrective bunches to be processed together. The sessions are flagged as “on going”
Production DB

4.1.2 Late calculation feature reference guide

This section explains the usage and behavior of the late calculation feature.

4.1.2.1 Build the XML input file

The Objective Status late calculation feature provide the ability to load new corrective events in the DataMart by the mean of an XML input file validated by the DTD `$TEMIP_SC_HOME/DTD/tsc_dm_ObjStatusCorrection.dtd` provided by SQM.

The events can:

- Replace the events occured between the start date and the end date of the correction: this is the Replace mode
- Being added to the events already existing between those dates: this is the Merge mode

Two types of message can be used:

Message 92: SMSCorrectionReq

This is the Service Monitoring Status Correction request which build corrective sessions for:

- Service Level Agreement (SLA) Level
- Service Component Instance (SCI) Level
- Service Parameter Objective (SP) Level

The following matrix shows which information must be provided depending on the chosen level.

	Service Level Agreement	Service Component Instance	Service Parameter Objective	Comment
`sd.name`	x	x	x	
`sla.name`	x	x	x	
`customer.name`	x	x	x	
`customerAggregate.flag`	x	x	x	True: if the SLA is Operational False: if the SLA is Contractual
`si.name`		x	x	
`scd.name`		x	x	
`sci.name`		x	x	
`rootComponent.flag`		x	x	True: the SCI is a Service False: the SCI is a Service Component
`sl.name`			x	
`csl.name`			x	
`slo.name`			x	
`parameter.name`			x	
`correction.startTimestamp`	x	x	x	
`correction.endTimestamp`	x	x	x	
`correction.mode`	x	x	x	`Merge/Replace`

The following requirements must be met to validate the correction.

- The correction range start date must be greater than the the creation timestamp of all mentioned entities (i.e. there cannot be any event on a Service Level Agreement before its creation time).
- The correction range end date must be lower or equal to the last summarized event.
- All mandatory information must be provided in the file.
- The XML file must be compliant with the DTD.
- The objective status values must be included between 0 and 1.

Here is a sample file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sc:SMSCorrectionReq SYSTEM
"DTD/tsc_dm_ObjStatusCorrection.dtd">
<sc:SMSCorrectionReq
xmlns:sc=”http://www.compaq.com/TeMIP/ServiceCenter”
msg.id="92"
      sd.name="Service1"
      sla.name="SLA1"
      customer.name="Customer1"
      customerAggregate.flag="False"
      correction.startTimestamp="2005-03-06T05:00:00.00"
      correction.endTimestamp="2005-03-07T05:00:00.00"
      correction.mode="Replace">
       <sc:CorrectiveObjectiveStatuses>
       <sc:CorrectiveObjectiveStatus
timeStamp="2005-03-06T05:00:00.00"
objectiveStatus="1.0"/>
</sc:CorrectiveObjectiveStatuses>
</sc:SMSCorrectionReq>
```

Message 93: SlaDutyCorrectionReq

This message gives the ability to add or remove “Out Duty” periods for a given Service Level Agreement. This will lead to recalculate Objective Status aggregegation at Service Level Agreement, Service Component Instance and Service Parameter Objective levels for the impacted Service Level Agreement.

The following matrix shows which information must be provided depending on the chosen level.

	Service Level Agreement	Comment
sd.name	x	
sla.name	x	
customer.name	x	
correction.startTimestamp	x	
correction.endTimestamp	x	
correction.mode	x	Merge/Replace

The following requirements must be met to validate the correction.

- The correction range start date must be greater than the the creation timestamp of all mentioned entities (i.e. there cannot be any event on a Service Level Agreement before its creation time).
- The correction range end date must be lower or equal to the last summarized event.
- All mandatory information must be provided in the file.
- The XML file must be compliant with the DTD.
- The objective status values must be comprised between 0 and 1.

Here is a sample file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sc:SlaDutyCorrectionReq SYSTEM
"DTD/tsc_dm_ObjStatusCorrection.dtd">
<sc:SlaDutyCorrectionReq
    xmlns:sc="http://www.compaq.com/TeMIP/ServiceCenter"
msg.id="93"
    sd.name="Service1"
    sla.name="SLA1"
    correction.startTimestamp="2005-04-02T04:00:00.00"
    correction.endTimestamp="2005-04-04T04:00:00.00"
    correction.mode="Replace">
  <sc:CorrectiveSlaDutyStatuses>
    <sc:CorrectiveSlaDutyStatus timeStamp="2005-04-
02T04:00:00.000" InDutyStatus="OutDuty"/>
    <sc:CorrectiveSlaDutyStatus timeStamp="2005-04-
03T00:00:00.000" InDutyStatus="InDuty"/>
    <sc:CorrectiveSlaDutyStatus timeStamp="2005-04-
03T04:00:00.000" InDutyStatus="OutDuty"/>
    <sc:CorrectiveSlaDutyStatus timeStamp="2005-04-
03T06:00:00.000" InDutyStatus="OutDuty"/>
    <sc:CorrectiveSlaDutyStatus timeStamp="2005-04-
04T00:00:00.000" InDutyStatus="InDuty"/>
  </sc:CorrectiveSlaDutyStatuses>
</sc:SlaDutyCorrectionReq>
```

4.1.2.2 Load the XML input file

The XML file is loaded by a specific loading tool provided with the DataMart. This tool can be used even if the DataMart component is not running because they do not interact. The production and staging databases must be running.

It is important to distinguish the correction loading which just consists in preparing and validating the corrective data, from the correction processing accomplished by the DataMart itself which consists in recalcultaing the Objective Status aggregations and integrating the corrective facts.

The loading tool is called via the command line, the user must be connected with the `sqmadm` login:

```
   $ ${TEMIP_SC_HOME}/bin/sqm_dm_load_os_correction
```

The options used by this tool are:

- `-load <file_name>`: loads the XML input file <file_name> containing the correction
- `-display`: list all the corrections and show their status

- `-v|version`: display the version of the tool
- `-h|help`: displays the online help

For example to load a file:

```
KSH> ${TEMIP_SC_HOME}/bin/sqm_dm_load_os_correction –load
myOsCorrection.xml
```

The output would be the following:

```
Processing file myOsCorrection.xml ...

Retrieving Staging connection string from Tibco
Repository...

Connecting to Staging database ...

Uploading request to Staging database ...

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sc:SMSCorrectionReq SYSTEM
"DTD/tsc_dm_ObjStatusCorrection.dtd">
<sc:SMSCorrectionReq
xmlns:sc="http://www.compaq.com/TeMIP/ServiceCenter"
msg.id="92"
      sd.name="Service1"
      sla.name="SLA1"
      customer.name="Customer1"
      customerAggregate.flag="False"
      correction.startTimestamp="2005-02-06T05:00:00.00"
      correction.endTimestamp="2005-02-07T05:00:00.00"
      correction.mode="Replace">
    <sc:CorrectiveObjectiveStatuses>
      <sc:CorrectiveObjectiveStatus timeStamp="2005-02-
06T05:00:00.00" objectiveStatus="1.0"/>
    </sc:CorrectiveObjectiveStatuses>
</sc:SMSCorrectionReq>


Reading reply from staging database...

<?xml version="1.0" encoding="UTF-8"?>
<sc:SMSCorrectionReply
xmlns:sc="http://www.compaq.com/TeMIP/ServiceCenter"
msg.id="92">
  <sc:Success>
    <sc:Message>Correction session with id 43 successfully
created</sc:Message>
    <sc:SuccessDetails>
      <sc:SuccessDetail/>
    </sc:SuccessDetails>
  </sc:Success>
</sc:SMSCorrectionReply>
Request committed ...
Disconnected from staging database...
Command terminated...
```

The status of the corrections can be checked by using the command:

```
KSH> ${TEMIP_SC_HOME}/bin/sqm_dm_load_os_correction
-display
```

The output would be the following:

```
Retrieving Staging connection string from Tibco Repository...

Connecting to Staging database...

SLA level corrective sessions:
+--+-----------+-------------------+-------------------+-------+----------+
|Id|SLA Name   |Start Date         |End Date           |Mode   |Status    |
+--+-----------+-------------------+-------------------+-------+----------+
| 4|SLA1O1G4C1i|2005-03-06 05:00:00|2005-03-07 05:00:00|Replace|Terminated|
|22|SLA1O4G3C1i|2005-04-10 00:00:00|2005-04-12 00:00:00|Replace|Terminated|
|23|SLA1O1G3C1i|2005-04-10 00:04:00|2005-04-12 00:04:00|Replace|Terminated|
|43|SLA1       |2005-02-06 05:00:00|2005-02-07 05:00:00|Replace|Terminated|
+--+-----------+-------------------+-------------------+-------+----------+
[5 rows of 6 fields returned]

SCI level corrective sessions:
+--+-----------+--------------+-----------+-------------------+-------------------+-------+----------+
|Id|SLA Name   |Parent SI Name|SI/SCI Name|Start Date         |End Date           |Mode   |Status    |
+--+-----------+--------------+-----------+-------------------+-------------------+-------+----------+
|26|SLA1O1G3C1i|SI1C          |S1_L2_1_i1 |2005-04-10 00:04:00|2005-04-12 00:04:00|Replace|Terminated|
+--+-----------+--------------+-----------+-------------------+-------------------+-------+----------+
[1 rows of 8 fields returned]

SP level corrective sessions:
+--+-----------+--------------+-----------+--------------+---------+-------------------+-------------------+-----+----------+
|Id|SLA Name   |Parent SI Name|SI/SCI Name|Parameter Name|SLO Name |Start Date         |End Date           |Mode |Status    |
+--+-----------+--------------+-----------+--------------+---------+-------------------+-------------------+-----+----------+
|27|SLA1O1G3C1i|SI1C          |S1_L2_1_i1 |IntP1         |S1_L2_1_1|2005-04-10 04:00:00|2005-04-12 04:00:00|Merge|Terminated|
+--+-----------+--------------+-----------+--------------+---------+-------------------+-------------------+-----+----------+
[1 rows of 10 fields returned]

SLA InDuty status corrective sessions:
+--+-----------+-------------------+-------------------+-------+----------+
|Id|SLA Name   |Start Date         |End Date           |Mode   |Status    |
+--+-----------+-------------------+-------------------+-------+----------+
| 1|SLA1O1G3C1i|2005-03-15 16:00:00|2005-03-24 16:48:42|Merge  |Terminated|
+--+-----------+-------------------+-------------------+-------+----------+
[5 rows of 6 fields returned]
```

The Status can be:

- Validated: the DataMart can process the correction
- Error: the DataMart encountered an error while processing the corrective session. (see DataMart's log)
- Ongoing: the DataMart is currently processing the corrective session.
- Terminated: the DataMart has processed the corrective session.

4.2 Performance Data Late Calculation

4.2.1 Overview of the performance data late calculation feature

The DataMart performance data late calculation feature gives the ability to modify aggregation results and raw data by integrating corrective measures to the existing DataMart data.

4.2.1.1 Architectural overview

In order to avoid concurrent access problems which could disturb the DataMart, the late calculation processing has been delegated to the DataMart itself by using a dedicated thread. The command line tool places the user's corrections in a queue to be processed by the DataMart either when invoking a specific AMI either at specified intervals depending on the Tibco configuration.

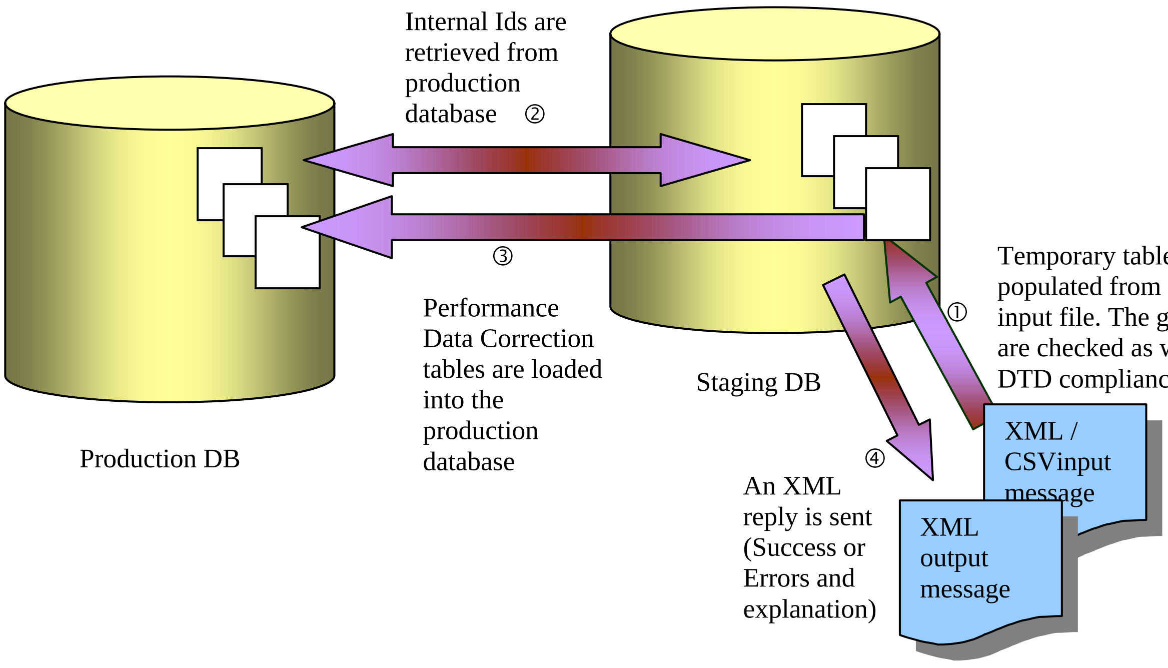
Step 1: load the corrective performance data.

The corrective performance data are provided by the user in XML or CSV files. These files are built by the user and loaded in the DataMart with the *sqm_dm_load_perf_correction* command line tool.

The command line tool will:

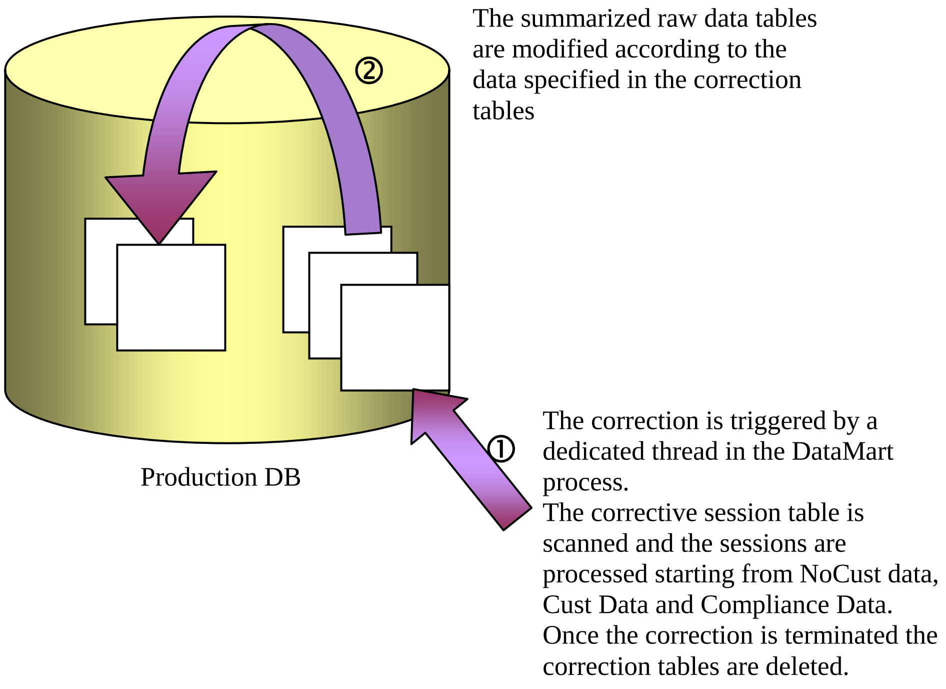
- Parse and validate the XML/CSV input files
- Populate temporary tables in staging
- Retrieve Internal IDs from the production database
- Check the dates validity
- Load correction tables into production database

The following schema shows the correction loading process:


Internal Ids are retrieved from production database ②
③
Performance Data Correction tables are loaded into the production database
Production DB
Staging DB
①
Temporary tables are populated from the given input file. The given dates are checked as well as the DTD compliance
④
An XML reply is sent (Success or Errors and explanation)
XML / CSVinput message
XML output message

Step 2: process the corrections

Once loaded and validated into the production database, the corrections are not integrated directly into performace data tables. The process of this correction can be either manual – using the "*startLatePerformanceCalculation*" AMI method through the Hawk Display tool – either automatically during the DataMart summarization process – when "Scheduled Late Performance Calculation" Tibco parameter is set to "true". In both cases, all pending corrective sessions are processed.


The summarized raw data tables are modified according to the data specified in the correction tables
②
Production DB
①
The correction is triggered by a dedicated thread in the DataMart process.
The corrective session table is scanned and the sessions are processed starting from NoCust data, Cust Data and Compliance Data.
Once the correction is terminated the correction tables are deleted.

4.2.2 Late calculation feature reference guide

This section explains the usage and behavior of the performance data late calculation feature.

4.2.2.1 Build the XML/CSV input files

The performance measure late calculation data have to be provided in an XML format containing all the corrective data or by the combination of an XML file containing a corrective header and a CSV file containing the corrective data.

A corrective session can:

- Replace the date occurred between the start date and the end date of the correction: this is the Replace mode
- Add to the corrective data to existing data between those dates: this is the Merge mode.
- Initialize a new component with provided data: this is the Initialize mode.

Since OpenView SQM 1.4, the datamart handles periods where no data is available for a parameter. It is of course possible to include such information in the late calculation input data using the “noValue” flag.

XML/CSV Sessions

This kind of corrective sessions use two input files:

- An XML file that describes the session. It contains only the “sc:PerfMeasureCorrectionReq” element with the following attributes:
    - msg.id : always ‘94’
    - sd.name: the impacted SD (ID of 16 characters). This attribute is present only for local SCD and must not be present for shared SCD.
    - scd.name: the impacted SCD (ID of 16 characters). This attribute is mandatory
    - correction.mode : ‘Replace’ | 'Merge' | ‘Initialize’
    - correction.startTimestamp and correction.endTimestamp: these attributes are mandatory and define the session boundaries. All corrective data have to be between correction.startTimestamp and correction.endTimestamp. The format is ‘YYYY-MM-DD’’T’’HH:MI:SS.MS’. For example: '2004-11-30T00:00:00'

Here is a sample XML file:

```
 <?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE sc:PerfMeasureCorrectionReq SYSTEM
"DTD/tsc_dm_PerfMeasureCorrection.dtd">
<sc:PerfMeasureCorrectionReq
        xmlns:sc =
'http://www.compaq.com/TeMIP/ServiceCenter'
        msg.id = '94'
        sd.name = 'SAMPLESD'
        scd.name = 'SAMPLESCD'
        correction.startTimestamp = '2004-11-30T00:00:00'
        correction.endTimestamp = '2005-01-01T23:59:00'
        correction.mode = 'Merge'
>
</sc:PerfMeasureCorrectionReq>
```

- A CSV file that contains all the corrective data. The format of this CSV file is: sci_name,param_name,value,no_value,timestamp[,customer].
  - Sci_id: the impacted sci name (ID of 16 characters).
  - Param_name: the impacted parameter (ID of 16 characters).
  - Value: the parameter value. The format depends on the parameter type. See below.
  - NoValue: the parameter no value flag (T/F).
  - Timestamp: the timestamp of the correction. The format is 'YYYY-MM-DD HH24:MI:SS.MS' (it doesn't contains the 'T' charactere as for XML format. Exemple: 2004-12-01 00:00:00
  - Customer: the impacter customer. This field has to be provided only for customer dependent parametets (ID of 16 characters).

  Here is an example that feeds SCI1 and SCI2 that contain several parameters (customer dependent and not dependent):

```
SCI1,YEAR,00,F,2004-12-01 00:00:00,C1
SCI1,WEEK,1,F,2004-12-01 00:00:00
SCI1,DAYNAME,WEDNESDAY,F,2004-12-01 00:00:00,C1
SCI1,HOUR,00,F,2004-12-01 00:00:00,C1
SCI1,MINUTES,12,F,2004-12-01 00:00:00
SCI1,DAY,01,F,2004-12-01 00:00:00
SCI1,SECONDDAY,00,F,2004-12-01 00:00:00,C1
SCI1,MONTHNAME,DECEMBER,F,2004-12-01 00:00:00
SCI1,TIMESTAMP,2004-12-01T00:00:00,F,2004-12-01 00:00:00
SCI1,RANDOM100,38,F,.19761437280289010951151954002350167 1,
2004-12-01 00:00:00,C1
SCI2,RANDOM1,F,.32344314184421777419956979388914966664,200
4-12-01 00:00:00
SCI2,TIME_S,'2004-12-01T00:00:00',F,2004-12-01 00:00:00,C1
SCI2,TIME_S,'2004-12-01T00:20:00',F,2004-12-01 00:20:00,C1
LATESCI,TIMESTAMP,2004-12-01T00:40:00,F,2004-12-01
00:40:00
```

  Note that for the TIMESTAMP parameter (which in our example has an AbsTime SQM datatype), the timestamp format of the measure (2004-12-01 00:40:00) is different from the value of the parameter which has to be in SQM format (2004-12-01T00:40:00).

XML Sessions

This kind of corrective sessions use only an XML input file that contains the session header and the corrective data.

The session header is the same has for XML/CSV session.

The corrective data are provided in the <sc:CorrectiveMeasures> element by a list of <sc:CorrectiveMeasure> elements.

Here is a sample XML file to illustrate the file content. The content of the <sc:CorrectiveMeasure> is detailed later.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE sc:PerfMeasureCorrectionReq SYSTEM
"DTD/tsc_dm_PerfMeasureCorrection.dtd">
<sc:PerfMeasureCorrectionReq
Removed for the sample (Same content as XML/CSV session)
```

```
>
 <sc:CorrectiveMeasures>
    <sc:CorrectiveMeasure sci.name="SCI1" >
      <sc:Measure timeStamp="2006-01-02T10:10:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">10</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T11:25:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">25</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T13:20:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">20</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T15:50:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">50</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T16:30:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">30</sc:ParameterValue>
      </sc:Measure>
    </sc:CorrectiveMeasure>
    <sc:CorrectiveMeasure sci.name="SCI2" >
      <sc:Measure timeStamp="2006-01-02T10:10:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">10</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T11:25:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">25</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T13:20:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">20</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T15:50:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">50</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T16:30:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES">30</sc:ParameterValue>
      </sc:Measure>
      <sc:Measure timeStamp="2006-01-02T17:30:00">
        <sc:ParameterValue datatype="Int"
parameter.name="MINUTES"
noValue=”True”></sc:ParameterValue>
      </sc:Measure>
    </sc:CorrectiveMeasure>
</sc:CorrectiveMeasures>
</sc:PerfMeasureCorrectionReq>
```

- <sc:CorrectiveMeasures>: No attribute, contains a list of <sc:CorrectiveMeasure> elements.
- <sc:CorrectiveMeasure>: Attributes : sci.name, customer.name. Contains a list of <sc:Measure>. Used to split data by SCI, Customer.
- <sc:Measure>: timestamp attributes. Contains a list of <sc:ParameterValue>. Split data by timestamp for a SCI(/Cust).

- <sc:ParameterValue>: contains parameter(s) value(s) for the SCI(/Cust)/Timestamp. Attributes: datatype, parameter.name, noValue.

Mesure Deletion

It is also possible to specify a list of deleted parameter for both XML and XML/CVS sessions.

If you specify some deleted parameters, all the data for these parameters (for specified SCI(/Cust)) will be deleted for the entire session boundary.

The deleted parameters are specified in the XML file. Here is an example of corrective session with deleted parameter:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE sc:PerfMeasureCorrectionReq SYSTEM
"DTD/tsc_dm_PerfMeasureCorrection.dtd">
<sc:PerfMeasureCorrectionReq
Removed for the sample
…
<sc:DeletionMeasures>
 <sc:DeletionMeasure sci.named='SCI1' customer.name='C1' >
  <sc:Parameter parameter.name='P1' />
  <sc:Parameter parameter.name='P2' />
 </sc:DeletionMeasure>
 </sc:DeletionMeasure sci.named='SCI1' customer.name='C2'>
  <sc:Parameter parameter.name='P1' />
 </sc:DeletionMeasure>
</sc:DeletionMeasures>
</sc:PerfMeasureCorrectionReq>
```

- **<sc:DeletionMeasures>:**
  Attributes: None.
  Nested Elements: <**sc:DeletionMeasure**>. This element contains the list of deleted parameters.

- <**sc:DeletionMeasure**>:
  Attributes: sci.name, customer.name (optional).
  Nested Elements: **<sc:Parameter>**. This element contains the list of the parameter to delete for the sci(/cust) **()**.

- **<sc:Parameter>**
  Attributes: parameter.name.
  This element defines the parameter to delete for the SCI(/Cust) defined in **<sc:DeletionMeasure>**.

Validation rules

The DataMart will load correction data after performing several validations on the input data. Here is the list of the validations performed:

- Mentioned entities (SD/SCD/SCI/Parameter/Customer) must exist in the DataMart.
- The correction range start date must be greater than the creation timestamp of all mentioned entities (i.e. there cannot be any parameter values on a SCI before its creation time).
- The XML file must be compliant with the DTD.
- The CSV file (if used) must respect the described format.
- The provided values must match their related SQM Datatype.
- Provided correction data timestamp must be between the session boundaries.

4.2.2.2 Load the XML input file

The XML file is loaded by a specific loading tool provided with the DataMart. This tool can be used even if the DataMart component is not running because they do not interact. The production and staging databases must be running.

It is important to distinguish the correction loading which just consists in preparing and validating the corrective data, from the processing of the correction accomplished by the DataMart itself which consists in recalcultaing the performance data aggregations and integrating the corrective facts.

The loading tool is called via the command line, the user must be connected with the `sqmadm` login:

```
  $ ${TEMIP_SC_HOME}/bin/sqm_dm_load_perf_correction
```

This tool loads a corrective performance measure values into the Datamart and provides follow up of their processing.

Options:

- -h|-help: Display help message.
- -v|-version: Display Tools and CLUI Versions.
- -display: lists the corrective sessions.
- -verbose: verbose mode to monitor the progress of the command execution.

Usage:

- sqm_dm_load_perf_correction -xmlfile <file_name>: loads the corrective performance measure values extracted from the xml file <file_name>. Xml file contains the corrective session header and corrective performance measures
- sqm_dm_load_perf_correction -xmlfile <file_name> -csvfile <file_name>: loads the corrective performance measure values from the CSV file <file_name>. Xml file contains only the corrective session header.

In cases of errors, some explanations on the problems detected are retuned.

In case of success, the following message is returned:

```
…
Reading reply from staging database...

<?xml version="1.0" encoding="UTF-8"?>
<sc:SMSCorrectionReply
xmlns:sc="http://www.compaq.com/TeMIP/ServiceCenter"
msg.id="92">
  <sc:Success>
    <sc:Message>Correction session with id 43 successfully
created</sc:Message>
    <sc:SuccessDetails>
      <sc:SuccessDetail/>
    </sc:SuccessDetails>
  </sc:Success>
</sc:SMSCorrectionReply>
Request committed ...
Disconnected from staging database...
Command terminated...
```

The status of the corrections can be checked by using the command:

```
KSH> ${TEMIP_SC_HOME}/bin/sqm_dm_load_perf_correction
–display
```

The output would be the following:

```
Retrieving Staging connection string from Tibco Repository...

Connecting to Staging database ...

corrective performance sessions :
+--+-------------+-------------------+-------------------+----+--------+
|Id|Scd Name     |Start Date         |End Date           |Mode|Status  |
+--+-------------+-------------------+-------------------+----+--------+
|1 |SCD1         |2004-01-01 05:00:00|2004-07-27 13:05:00|Init|On going|
|2 |SCD2         |2004-01-01 05:00:00|2004-07-27 13:05:00|Init|Loaded  |
+--+-------------+-------------------+-------------------+----+--------+
[2 rows of 6 fields returned]

Disconnected from staging database...
```

The Status can be:

- Loaded: the DataMart can process the correction
- Load pending : the corrective session is being loaded
- Ongoing: the DataMart is currently processing the corrective session.
- Terminated: the DataMart has processed the corrective session.

The Status can be:

- Validated: the DataMart can process the correction
- Error: the DataMart encountered an error while processing the corrective session. (see DataMart’s log)
- Ongoing: the DataMart is currently processing the corrective session.
- Terminated: the DataMart has processed the corrective session.

4.2.2.3 Processing correction

Once loaded into the production database, the corrections are not integrated directly into performace data tables.

The processing of these corrections can be triggered manually, using the “startLatePerformanceCalculation” AMI of the DataMart micro-agent. This AMI can be called via a script or via the Tibco Hawk Display GUI.

If you intend to use the AMI in a script, you can use the following command:

```
temip_sc_selfmgmt    -daemon
$TEMIP_SC_REPOSITORY_SESSION_DAEMON \
                    -network
"$TEMIP_SC_SELF_MGMT_RV_NETWORK"  \
                    -service
$TEMIP_SC_SELF_MGMT_RV_SERVICE \
                    -ha `hostname`-HA \
                    -hma
${TEMIP_SC_KERNEL_ID}_slreporting_DataMart_MA \
                    -m startLatePerformanceCalculation
```

Note that the value of –ha and –hma to provide may be different. They depend on your kernel configuration.

The corrective session can also be triggered automatically as part of the summarization process. To do this, you have to set the “Scheduled Late Performance Calculation” Tibco parameter to “true” using the Tibco designer and restart the DataMart.

In both case, when correction is triggered, all pending correction sessions are processed one by one in the same order as the loading.

The tables impacted by a corrective session are:

- SCI_xxxxx_VRD: Raw Data in vertical format.
- SCI_xxxxx_HRD: Raw Data in horizontal format (grouped by SCI/(Cust)).
- SCI_xxxxx_H|D|W|M|Q|Y: Aggregated Data in horizontal format.

4.2.2.4 Examples

This section provides few examples of the usage of late corrections.

For each example, data before the correction, corrective data and data after the correction are described. To simplify the example, only one SCI is impacted, but this is not a limitation. Also, input data are provided in XML/CSV format.

Initialize

We have created a SCI one month ago but no data have been collected due to a wrong DFI binding. It is possible to submit an Initialize corrective session for this SCI.

Corr1.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE sc:PerfMeasureCorrectionReq SYSTEM
"DTD/tsc_dm_PerfMeasureCorrection.dtd">
<sc:PerfMeasureCorrectionReq
        xmlns:sc = 'http://www.compaq.com/TeMIP/ServiceCenter'
        msg.id = '94'
        sd.name = 'SAMPLESD'
        scd.name = 'SAMPLESCD'
        correction.startTimestamp = '2006-01-01T00:00:00'
        correction.endTimestamp = '2006-02-01T23:59:00'
        correction.mode = 'Initialize'
>
</sc:PerfMeasureCorrectionReq>
```

Corr1.csv

```
SC1,P1,1, F,2006-01-01 00:00:00
SC1,P1,2, F, 2006-01-01 01:00:00
SC1,P1,3, F, 2006-01-01 02:00:00
…
SC1,P1,23434,F, 2006-02-01 23:59:00
```

```
temip_sc_load_perf_correction –xmlfile corr1.xml –csvfile corr1.csv
```

My SCI has another parameter that I forget to initialize. I can submit a new Initialize session for the SCI with the values for P2.

Corr2.csv

```
SC1,P2,-10,F,2006-01-01 00:00:00
SC1,P2,-20, F,2006-01-01 01:00:00
SC1,P2,-30, F,2006-01-01 02:00:00
…
SC1,P2,-23434, F,2006-02-01 23:59:00
```

```
temip_sc_load_perf_correction –xmlfile corr1.xml –csvfile corr2.csv
```

Note that the same XML file is used.

I did a mistake during extraction of P2, it is possible to correct these values with a Replace mode session.

Corr3.xml: Same a Corr1.xml but correction.mode = 'Replace'

Corr3.csv: Same a Corr2.csv but with correct values this time.

```
temip_sc_load_perf_correction –xmlfile corr3.xml –csvfile corr3.csv
```

In fact, the 1 hour sampling is not enough. We want 1 value each ½ hour. You can choose to recreate all new sample data and submit them in Replace mode, but you can also provide the missing sample and submit them in ‘Merge Mode.

Corr4.xml: Same a Corr1.xml but correction.mode = ‘Merge'

```
SC1,P1,1.5,F, 2006-01-01 00:30:00
SC1,P2,15, F, 2006-01-01 00:30:00
SC1,P1,2.5, F, 2006-01-01 01:30:00
SC1,P2,25, F,2006-01-01 01:30:00
…
SC1,P1,14454.5, F, 2006-02-01 23:30:00
SC1,P2, , T, 2006-01-01 23:30:00
```

Note that P1 and P2 are provided in the same corrective session.

Appendix A

Service Health Indicators computation – SA%, MTBF, MTTR

This appendix details the calculation of the Service Health Indicators within Service Quality Manager: SA%, MTBF, MTTR.

Service Availability – SA%

The standard formula is used, adapted from the SA formula defined in *TMF701, Performance Reporting Concept & Definitions V1.1* making use of the Objective Status (OS) instead of the Service Degradation Factor (SDF):

$$SA = \frac{\sum(OS * \Delta T)}{\sum(\Delta T)}$$

*OS*Delta T* is the multiplication of each OS per its duration. It is divided by the observation duration, which depends on the aggregation granularity (month, year and so on).

This formula is impacted by the "In Service Hours" because the cumulated duration is increased only if the SLA status is equal to *InDuty* for the currently processed time reference.

The cumulated duration is reset at the beginning of each reference period.

Mean Time Between Failure – MTBF

The standard formula, derived from *TMF701, Performance Reporting Concept & Definitions V1.1*, is:

$$MTBF = \frac{\Delta T * number(objects)}{number(FailedPeriods)}$$

Delta T is the observation duration, which depends on the aggregation granularity (month, year and so on).

Considering that in the case of Service Quality Manager, the health indicators are computed each time for a single object (parameter, SI, SCI and so on), the formula becomes:

$$MTBF = \frac{\Delta T}{number(FailedPeriods)}$$

This formula is impacted by the "In Service Hours" because the number of failed periods is incremented only if the SLA has its status equal to *InDuty*.

The number of failed periods is reset at the beginning of each reference period.

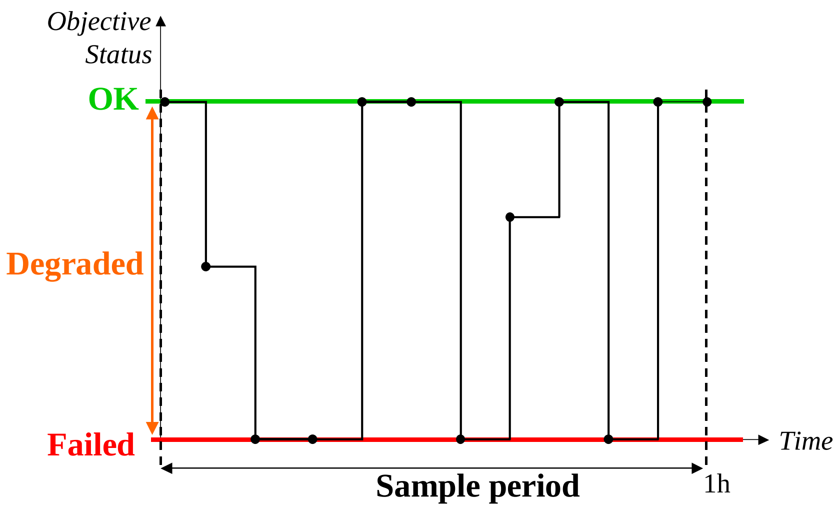
Warning

Anything that happens before or after the considered period is not taken into account:

- If the considered period starts with a failed state, this time is considered as the failed state beginning.
- If the considered period ends with a failed state, this time is considered as the failed state ending.

The following example gives a visual interpretation of the MTBF:

Example 1 MTBF Computation



In this example, the MTBF is:

```
MTBF = (sample period in hours)/ (number of Failed periods)
     = 1/3 hour
```

Mean Time To Repair – MTTR

The standard formula is:

$$MTTR = \frac{\sum(FailedDuration)}{number(FailedPeriods)}$$

This formula is impacted by the "In Service Hours" because the number of failed periods is incremented only if the SLA has its status equal to *InDuty*. In the same way, the cumulated failed duration is incremented only if the SLA has its status equal to *InDuty*.

The number of failed periods is reset at the beginning of each reference period.

The cumulated failed duration is reset at the beginning of each reference period.

Warning

As for the MTBF computation, anything that happens before or after the considered period is not taken into account.

Example 2 MTTR Computation

Continuing with the example for the MTBF calculation, the MTTR equals:

```
MTTR = (number of hours in Failed state) /
       (number of Failed periods)
     = 4 / 3
     = 1.33 hours per corrective maintenance action
```

Cumulated OK / Degraded / Failure Duration

The cumulated OK (respectively degraded, failure) duration is incremented only when the SLA status is set to InDuty for the currently processed time reference.

The cumulated OK (respectively degraded, failure) duration is reset at the beginning of each reference period.

Appendix B

Production Area Database Model

This chapter lists all the tables that are filled by the DataMart Migration Component, which are present in the Production Area.

Table 2 Figures Keys

Key	Meaning
CP	Primary Key
I	Index
CE	Foreign Key
BOLD	The column is mandatory, that is, cannot be equal to NULL

The following table fields can be found in several table definitions (dimensions or facts). They are not intended for the end user (they are used by the DataMart Migration Component):

- GLOBAL_NAME.
- IS_SUMMARIZED.

Static Model - Generic Data

Dimensions

This section provides a listing of all the dimensions that the Service Quality Manager DataMart fills and handles.

The majority of the dimension table fields are retrieved exactly from the Service Quality Manager management model definition hence need not be explained specifically.

Note

All the dimensions have a column that is named *IS_MARKED_AS_DELETED* indicating whether the object has been removed from the model. Indeed it cannot be actually removed from the table because of summarized statistics that provide data on formerly used objects. See Section 2.1.2.1.

Time Dimensions

The possible values for the Time Dimension fields are given in the following table. The values may be changed while generating the time dimension.

Table 3 Time Dimensions Fields Possible Values

Field	Possible Values (default)
*_OVERALL	Computed since the 1st January 1970.
DAY_NAME	"Monday", "Tuesday", …
DAY_MONTH_NUMBER	1..12
DAY_WEEK_NUMBER	1..7
DAY_YEAR_NUMBER	1..366
MINUTE_ID	0..59
FIVE_MINUTE_ID	1..12
FIFTEEN_MINUTE_ID	1..4
HOUR_ID	0..23
MONTH_NAME	"January", "February", …
MONTH_NUMBER	1..12
WEEK_NUMBER	1..53
QUARTER_NAME	"Q1", "Q2", …
QUARTER_NUMBER	1..4
SEMESTER_NAME	
SEMESTER_NUMBER	1..2
YEAR_NUMBER	The possible values depend on the input data while configuring the Service Quality Manager DataMart. For example: 2003, 2004, …

As described in the *OpenView Service Quality Manager DataMart Configuration and Administration Guide*, the DESCRIPTIONs, IS_HOLIDAY and EVENT fields are user-defined and can be filled while configuring the Service Quality Manager DataMart.

Time Dimension

The Time Dimension (TIME_DIM) contains the time identifiers that are used by all the fact tables.

Hourly Time Dimension

This dimension (HOURLY_TIME_DIM) is used in the hourly summarization tables.

Daily Time Dimension

This dimension (DAILY_TIME_DIM) is used in the daily summarization tables.

Weekly Time Dimension

This dimension (WEEKLY_TIME_DIM)is used in the weekly summarization tables.

Monthly Time Dimension

This dimension (MONTHLY_TIME_DIM) is used in the monthly summarization tables.

Quarterly Time Dimension

This dimension (QUARTERLY_TIME_DIM) is used in the quarterly summarization tables.

Yearly Time Dimension

This dimension (YEARLY_TIME_DIM) is used in the yearly summarization tables.

Customer Dimension

This dimension (CUSTOMER_DIM) contains the data characterizing a Customer.

Service Definition, Service Component Definition Dimension

This dimension (SD_AND_SCD_DIM) contains the data characterizing a Service Definition or a Service Component Definition.

The SD_SCD_LONG_PREFIX field is used to build the related table name where the performance data is stored for the relevant SD / SCD.

The SD_SCD_SHORT_PREFIX field is for internal use only.

Service Instance / Service Component Instance Definition

This dimension (SI_AND_SCI_DIM)contains the data characterizing a Service or a Component.

The SD_SCD_PREFIX field is used to build the related table name where the performance data is stored for the relevant SD / SCD.

The CA_NAME field is defined only when a performance data has been processed for the relevant SI / SCI.

Service Instance Group Dimension

This dimension (SIG_DIM)contains the data characterizing a SIG.

Service Level Agreement Dimension

This dimension (SLA_DIM) contains the data characterizing an SLA.

Service Level Dimension

This dimension (SL_DIM) contains the data characterizing a Service Level.

Service Level Objective Dimension

This dimension (SLO_DIM) contains the data characterizing an SLO.

Parameter Dimension

This dimension (PARAMETER_DIM) contains the data characterizing a Parameter.

Property Dimension

This dimension (PROPERTY_DIM) contains the data characterizing a Property.

Enumeration Dimension

This dimension (ENUM_DIM) contains the Enumeration string labels for all the parameters and properties.

Label Dimensions

Administrative State Label Dimension

This dimension (ADMINISTRATIVE_STATE_LBL_DIM) contains the Enumeration string labels for the Administrative State.

The possible values for the LABEL are:

- Locked
- Unlocked

Availability Status Label Dimension

This dimension (AVAILABILITY_STATUS_LBL_DIM) contains the Enumeration string labels for the Availability Status.

The possible values for the LABEL are:

- Available
- InTest
- Failed
- PowerOff
- OffLine
- OffDuty
- Dependency
- Degraded
- NotInstalled
- Logfull
- Unknown

Crossed Param Status Label Dimension

This dimension (CROSSED_PARAM_STATUS_LBL_DIM) contains the possible values for the Crossed Parameter (used in the crossed_parameter fact tables).

The possible values for the LABEL are:

- Start
- End

Operational State Label Dimension

This dimension (OPERATIONAL_STATE_LBL_DIM) contains the Enumeration string labels for the Operational State.

The possible values for the LABEL are:

- Disabled
- Enabled
- Unknown

Quality of Service Label Dimension

This dimension (QOS_STATUS_LBL_DIM) contains the possible values for the SLO Quality of Service.

The possible values for the LABEL are:

- Increasing
- Decreasing

Update Flag Label Dimension

This dimension (UPDATE_FLAG_LBL_DIM) contains the possible string values for an update event.

The possible values for the LABEL are:

- Added
- Updated
- Deleted
- None

Managed Object Dimension

This table is deprecated with this new version of the Service Quality Manager DataMart.

Objective Threshold Dimension

This dimension (OBJECTIVE_THRESHOLD_DIM) contains the data characterizing an Objective Threshold.

Compliance Service Level Objective Dimension

This dimension (CVL_SLO_DIM) contains the data characterizing a Compliance SLO.

Compliance Parameter Dimension

This dimension (CVL_PARAMETER_DIM) contains the data characterizing a Compliance Parameter.

Compliance Objective Threshold Dimension

This dimension (CVL_OBJECTTIVE_THRESHOLD_DIM) contains the data characterizing a Compliance Objective Threshold.

Configuration

This table (SC_CONFIGURATION) contains a set of configuration variables that are helpful for the DataMart, but that could also be used by a reporting function.

This table has no primary key.

Table 4 Configuration Data Stored in the SC_CONFIGURATION Table

NAME	DESCRIPTION / Example
Data Collection Interval	Retrieved from the SQM platform configuration (for example, "5").
DataMart Granularity	Retrieved from the DataMart Migration Adapter configuration (for example, "5").
Degradation State Level	Retrieved from the SQM platform configuration (for example, ".8").

NAME	DESCRIPTION / Example
Violation State Level	Retrieved from the SQM platform configuration (for example, ".2").

Dynamic Model - Specific Data

The dynamically created dimension and fact table names are generated using the Service Definition name field, and if necessary the Service Component Definition name field.

The following name construction rule is used for the SCI level tables. The same rule is used for the compliance level tables, 'SCI’ being replaced by 'CVI’.

Specific performance data fact table name:

SCI_<SD name>[_<SCD name>[_<order number on 1 digit>]]_dat

Specific summarized performnace data fact table name:

SCI_<SD name>[_<SCD name>][_<order number on 1 digit>]] [_h/_d/_w/_m/_y]

Specific summarized performance data fact view name (provides a unique view for all summarization level)

SCI_<SD name>[_<SCD name>][_<order number on 1 digit>]]_V

In the case of a shared SCD between several SD, the measures are put in a unique table named : SCI_ND$_<SCD name>_dat

Appendix C

Glossary

This glossary defines the terminology commonly used in OpenView Service Quality Manager.

Auto instantiate (SLA Administration)

This action automatically creates an Instance of the Object selected. When the instance is created, the initial values of its instance variables are assigned.

BI

See Business Intelligence.

Business Intelligence (BI)

A broad category of applications and technologies for gathering, storing, analyzing, and providing access to data that helps users make better business decisions.

Collected binding

Describes how *collected parameters* are filled from *measurement parameters*: either directly assigned or through a more complex expression.

Collected parameters

Known as KPI in the TMF, they represent the parameters collected from the Service Adapters (*measurement parameters*) and mapped into SQM *service component* parameters.

Computed binding

Describes how *computed parameters* are filled from *collected parameters*: either directly assigned or through a more complex expression.

Computed parameters

Known as KQI in the TMF, they represent the parameters calculated from *collected parameters*.

CNM

See Customer Network Management

Customer

Companies or organizations that make use of the *services* offered by a *service provider*, based on a contractual relationship.

Customer Network Management

Customer network management is enabled by means of tools that provide business customers with access to management information originating from the service provider.

Data collection interval

The interval of time over which performance parameters are retrieved from monitored service resources. This interval does **not** have to be the same as the *measurement interval* because *service adapters* or service resources may buffer statistics.

Data feeder

OpenView Service Quality Manager’s source of data. A data feeder models service resources by defining one or more service parameters.

Data feeder definition

The static definition of a data feeder that models service resources by defining one or more service parameters.

Degraded service

The presence of anomalies or defects that cause degradation of the *quality of service*, but do not result in total failure of the *service*.

Instantiate (SLA Administration)

Instantiate differs from Auto Instantiate in that items are instantiated individually.

Measurement interval

The time interval over which each service parameter is measured. For example, a parameter may be the number of discarded packets, measured over a 15-minute measurement interval.

Measurement parameters

They represent the parameters directly collected by the Service Adapters. These parameters are defined in the *Data Feeders*.

Measurement Reference Point (MRP) naming scheme

This is the formal description of how the measurement point name is built, that is, by concatenating the values of Data Feeder properties and fixed strings.

Mobile Virtual Network Operator

A mobile operator, which does not own its own spectrum and usually does not have its own network infrastructure. Instead, MVNOs have business arrangements with traditional mobile operators to buy **minutes of use** for sale to their own customers.

MRP

See Measurement Reference Point.

MVNO

See Mobile Virtual Network Operator.

Parameter

A value or set of values that are periodically updated and that help determine the quality of service.

Parameter objective

A set of objectives for the parameters belonging to a service.

Property

Special static parameters that are given a value only when an instance of an OpenView Service Quality Manager **Object** is created. For example, a Service Component can have a property called "location".

QoS

See Quality of Service.

Quality of Service (QoS)

The ITU-T has defined Quality of Service as "the collective effect of service performances that determine the degree of satisfaction of a user of the service".

Service

A Service is a set of independent functions (Service Components) that consist of hardware and software elements and an underlying communications medium. A Service can include anything from a single leased-line service, to a complex application, such as vision conferencing.

Service availability

A measurement made in the context of a *service level agreement* that is expressed as a percentage. This percentage indicates the time during which the *service* is operational at the respective *service access points*.

ServiceCenter Repository

The ServiceCenter Repository is the storage center for all Service Quality Manager data. It receives data from the various Service Quality Manager interfaces and each interface can request information from the Repository.

Service component

An independent function that is part of a *service*, such as a hardware or software element, or the underlying communications medium.

Service component instance

The instance of a Service Component Definition that is active in the network, such as an instance of the IPAccess Service Component definition called "pop".

Service Level (SL)

Defines Service Parameters and operational data enforced by the Service Level Agreement (for example, Max Jitter < 10 ms).

Service Level Agreement (SLA)

There are two type of Service Level Agreement, the **Customer** Agreement: a contract between a *service provider* and a *customer*, which specifies in measurable terms what the service provider supplies to its customers, and the Operational Service Level Agreement, which specifies in measurable terms the operational levels of the Service. A *service level agreement* is composed of individual objectives.

Service Level Objective (SLO)

The set of objectives for the parameters belonging to a Service or Service Component.

Service parameter

See *parameter*.

Service provider

A company or organization that provides *services* as a business. Service providers may operate networks or may integrate the services of other providers.

Service Instance (SI)

The instantiated service definition that is active in the network, such as an instance of the video service definition called "Paris".

Service Instance Group (SIG)

A **group** of *service instances* against which the *service availability* must be reported. Each *service instance* belongs to one or more Service Instance Groups and each SIG contains at least one Service Instance. The relationship between the SIG and the Service Instances is defined in their *service level agreement*.

Service quality parameters

They represent *computed* and *collected* parameters

SI

See Service Instance.

SIG

See Service Instance Group.

SL

See Service Level

SLA

See Service Level Agreement.

SLO

See service level objective.

Subscriber

The entity responsible for the payment of charges incurred by one or more users.

User

An entity designated by a customer to use the services of a telecommunication network, such as a person using a UMTS mobile station as a portable telephone.

Index

A

Associations 24

C

Conversions
  Datatypes 14
Cumulated Degradation Duration 20
Cumulated Degraded Duration 47
Cumulated Failure Duration 20, 47
Cumulated OK Duration 20, 47

D

Data Warehouse 7, 9
DataMart 7
DataMart Granularity 13, 52
DataMart Migration Component 11
  Architecture 11
Dimension Tables 23
Dynamic Dimensions 12

F

Fact Tables 23

G

Global Name 14
Granularity 8

I

Inventory History 23
IS_MARKED_AS_DELETED 13

M

Mean Time Between Failure 45, *Voir*
Mean Time To Repair 20, 47
Measure
  Duplicates 22
Measure
  Late 22
MTBF *See* Mean Time Between Failure
MTTR *See* Mean Time To Repair

O

Objective Status 18
OLAP 7, 9

P

Production Area 11
Production Area Management Mode
  Delete, Move, NoAction 21

S

Service Availability 45
Service Availability Percentage 19
Service Degradation Factor 18
Staging Area 11
Staging Session 12
Summarization
  Maximum Value 16, 17
  Minimum Value 16, 17
  Relative Time 18
Summarization
  Average Value 16
Summarization
  Performance data 17
Summarization
  Average Value 17
Summarization
  Average Value 17
Summarization
  Enumerated Values 17
Summarization
  Absolute Time 18

T

Time Dimension 13
  Summarized 15
Time Dimensions 49
Time Identifier 13
  Summarized 15
Time Zone 14
TIME_REF 23

invent

Java™ COMPATIBLE