hp OpenView
Service Quality Manager

Overview

Edition: 1.2

for the HP-UX and Microsoft Windows Operating Systems

March 05

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company

United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

HP-UX Release 10.20 and later and HP-UX Release 11.00 and later (in both 32 and 64-bit configurations) on all HP 9000 computers are Open Group UNIX 95 branded products.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Netscape is a U.S. trademark of Netscape Communications Corporation.

NMOS™ is a trademark of RiverSoft Technologies Limited.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Oracle7™ and Oracle7 Server™ are trademarks of Oracle Corporation, Redwood City, California.

PostScript® is a trademark of Adobe Systems Incorporated.

Riversoft™ is a trademark of RiverSoft Technologies Limited.

UNIX® is a registered trademark of The Open Group.

Windows® and Windows NT® are U.S. registered trademarks of Microsoft Corporation.

X/Open® is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Contents

Preface

This document is a global overview of the HP OpenView Service Quality Manager (SQM). OpenView SQM monitors the quality of service and service level agreements. Service providers can use it to pro-actively manage their services and to prioritize the actions taken depending on the customer's rate of payment.

This document provides a general technical overview of OpenView SQM and provides sample case studies.

Intended Audience

This document addresses service designers, service operators, and anyone involved in service monitoring and reporting that needs a general overview of OpenView SQM.

Required Knowledge

Chapter 4 describes the OpenView SQM object model and assumes that you understand the basics of UML. For a quick overview of UML, refer to Appendix C.

Supported Software

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

Product Version	Operating System
OpenView Service Quality Manager 1.2	HP-UX 11.11 Windows XP

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.
- Path names
- 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 OpenView SQM documentation set includes the following:

- *OpenView SQM SLA Monitoring UI User's Guide*
- *OpenView SQM Service Designer UI User's Guide*
- *OpenView SQM SLA Administration UI User's Guide*
- *OpenView SQM Overview*
- *OpenView SQM Getting Started Guide*
- *OpenView SQM Information Modeling Reference Guide*
- *OpenView SQM Installation Guide*
- *OpenView SQM Administration Guide*
- *OpenView SQM Reference Guide for Oracle Use*
- *OpenView SQM Datamart User's Guide*
- *OpenView SQM Reporting Customization and User's Guide*
- *OpenView SQM Gateway for OpenView TeMIP Fault Management Installation and User's Guide*
- *OpenView SQM Gateway for OpenView Operations Installation and User's Guide*

Refer to the following document for useful reference information:

- *TeleManagement Forum Service Level Agreement Management Handbook, v 1.5.*
- *Wireless Service Measurements Handbook GB923 version 1.5 from the TeleManagement Forum.*

Support

Please visit our HP OpenView web site at: HP OpenView

There you will find contact information as well as details about the products, services, and support OpenView has to offer.

The "hp OpenView support" area of the OpenView web site includes:

- Documentation you can download
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

Chapter 1

Introduction to Service Quality Management

In today's competitive market, service quality management is now widely recognized as a high priority for service providers. At the same time, services are becoming increasingly complex. By controlling the quality of business-critical services and offering attractive service level agreements, service providers keep customers and ensure customer loyalty.

The design of OpenView Service Quality Manager (SQM) addresses the challenges of service quality management faced by service providers in today's market.

OpenView SQM answers the demands of end-to-end service quality management with tools for managing quality of service and service level agreements. It allows service providers to move towards proactive service management while focusing on their highest paying customers.

This chapter introduces the motivations and requirements for service quality management in the following sections:

- Section 1.1:What is a Service?
- Section 1.2: What is Service Quality?
- Section 1.3: About Service Quality Management
- Section 1.4: About Service Level Agreements

1.1 What is a Service?

A service is a commercial offer sold to a customer. In the service provider's environment, the scope of a service definition is very large. The definition includes not only the pricing characteristics, but also a thorough description of the infrastructure and technologies used, and the configuration of the equipment that provides the service to the customer.

In OpenView SQM:

- A service definition is dedicated to service quality management and contains any information useful for evaluating service quality.
- You can model any service no matter what its level of complexity.
- You can relate a service to one customer, several customers, or no customers at all.

The following sections describe how OpenView SQM defines services, models complex services, and relates services to customers.

1.1.1 Defining Services for Service Quality Management

Because OpenView SQM is a service quality management system, the service definition provides a service level management focus on service quality metrics. As a result, a service definition contains any information used for end-to-end service quality evaluation, such as quality of service parameters, quality of end-user experience parameters, and quality of business parameters.

1.1.2 Modeling Complex Services

Services are usually complex and can be broken down into service components. A service includes anything from a single leased-line service, to a complex application, such as video conferencing or basic network services like e-ticketing. For complex services, OpenView SQM groups all the information from the various service components and provides an end-to-end analysis of the service quality.

1.1.3 Relating Services to Customers or Operations

OpenView SQM was designed to provide flexible service definitions that can relate services to one customer, several customers, or no customers at all. OpenView SQM provides three different types of services:

- Single-customer services.

  These services are associated with a single known customer, such as fixed services (like synchronous digital and ADSL) and mobile services (like WAP over GPRS).

- Multi-customer services.

  Various customers share these services. A football score service that provides personalized content to individual subscribers is an example of a multi-customer service.

- No-customer services.

  Various customers share these services, but their identities are unknown or not needed for service quality evaluation. For example, the customer identity is lost when an ADSL customer uses an IP service with an IP address assigned by a dynamic host configuration protocol (DHCP) server during the connection phase.

1.2 What is Service Quality?

Customers express their perception of service quality as a degree of satisfaction. So, assessing service quality implies taking into account various types of quality measures, including quality of service (QoS), quality of experience (QoE), and quality of business (QoBiz).

Key quality indicators (KQI) come from all the different types of quality measures so that you can evaluate the service quality as a whole.

OpenView SQM brings together all of these service quality concepts to provide a thorough evaluation of your services. The following sections describe the key types of service quality measure.

1.2.1 Quality of Service

The quality of service (QoS) is characterized by technical performance parameters understood by both customers and service providers. These parameters are generally measured at the infrastructure level and are highly dependant on the underlying technologies of the service. QoS parameters include bandwidth, service availability, and maximum authorized packet or cell loss rate.

The level of quality is expressed through values attributed to the QoS parameters.

A key performance indicator (KPI) gives information about the behavior of the infrastructure equipment involved in providing the service. KPIs include CPU load, number of resets, and downtime.

1.2.2 Quality of Experience

The quality of experience (QoE) is the level of customer satisfaction while using the service, or while in contact with the service provider. It can be difficult to evaluate the perception of quality from the customer point of view. Service providers use means such as the following to measure the quality of experience:

- Usage

  The usage parameters can provide hints about customer satisfaction during the use of a service. Usage parameters include period of customer activity during the day and the mean time of service use. However, a high usage does not necessarily show that the customer is satisfied with the service. For example, the customer may be struggling to make a transaction because the network is slow.

- End-user behavior simulator

  Because it is difficult to know the customer's QoE while using the service, service provider's use end-user behavior simulators to evaluate the service at different times of the day. Service providers generally distribute these systems in several places. They play the role of end-users executing pre-defined generic transactions, while measuring QoS parameters. End-user behavior simulators measure parameters such as time to connect to the server and time to download information. These measures are not specific to a customer, but instead give a global view of the service.

- Performance of customer relationship processes

  Service providers evaluate the QoE during contact with the customer by measuring the performance of the different customer relationship processes within its organization. For example, the service provider can measure the time it takes the help desk to respond on the phone and the time it takes to repair the service.

- Surveys

Service providers also use customer surveys and customer comments to assess the customer perception of service quality.

1.2.3 Quality of Business

The quality of business (QoBiz) is the business achievement compared to the costs incurred to run a service. The QoBiz is primarily of interest to the service provider. A service is valuable if it brings revenues, so the QoBiz concerns itself with measures like the number of customers buying the service compared to the amount of equipment deployed, the number of connections each day on a specific web page, and the number of new customers each month for a particular service.

1.3 About Service Quality Management

Service quality management is a key to evaluating and controlling the customer perception of the quality of service. Monitoring and managing service quality bridges the gap between the customer perception of the service and the equipment performance measured by service providers.

An efficient system of service quality management should express service level agreements in terms that are easy for you and your customers to understand and measure.

This section describes the following aspects of service quality management:

- What is service quality management?
- Managing service quality for complex new services.
- Benefits of service quality management.

1.3.1 What is Service Quality Management?

To manage service quality efficiently, you need information about:

- Service quality at any time, in real-time.
- Service quality over time for business critical services.

Above all, service providers must be able to aggregate this information and draw conclusions rapidly so that they can take steps to correct service quality before the service degrades and a service level agreement violation occurs.

As a result, service quality management consists of:

- Collecting and storing relevant key quality indicators, such as quality of service parameters, quality of experience parameters, usage parameters, and performance parameters from the service provider’s processes.
- Analyzing the following key quality indicators in real-time:

  Evaluating the overall quality delivered from an end-to end service perspective.

  Establishing whether service quality conforms to the service level agreement (SLA) and identifying if service degradation is in the process of causing an SLA violation.

  Rapidly publishing the results of the analysis to the relevant staff, such as network operators, service operators, and management operators.
- Providing reports that service managers can use to analyze trends and verify the customer service quality.

The design of OpenView SQM addresses these tasks, helping you efficiently manage your service quality and carefully monitor your service level agreements.

1.3.2 Managing Service Quality for Complex New Services

New services, such as high speed broadband access, mobile access (UMTS, GPRS), and other enabling value-added services, rely on a heterogeneous environment composed of wireline and wireless networks, IT infrastructure (including machines, databases, and servers), and other applications.

Traditional network management systems have the following limitations in managing these complex services:

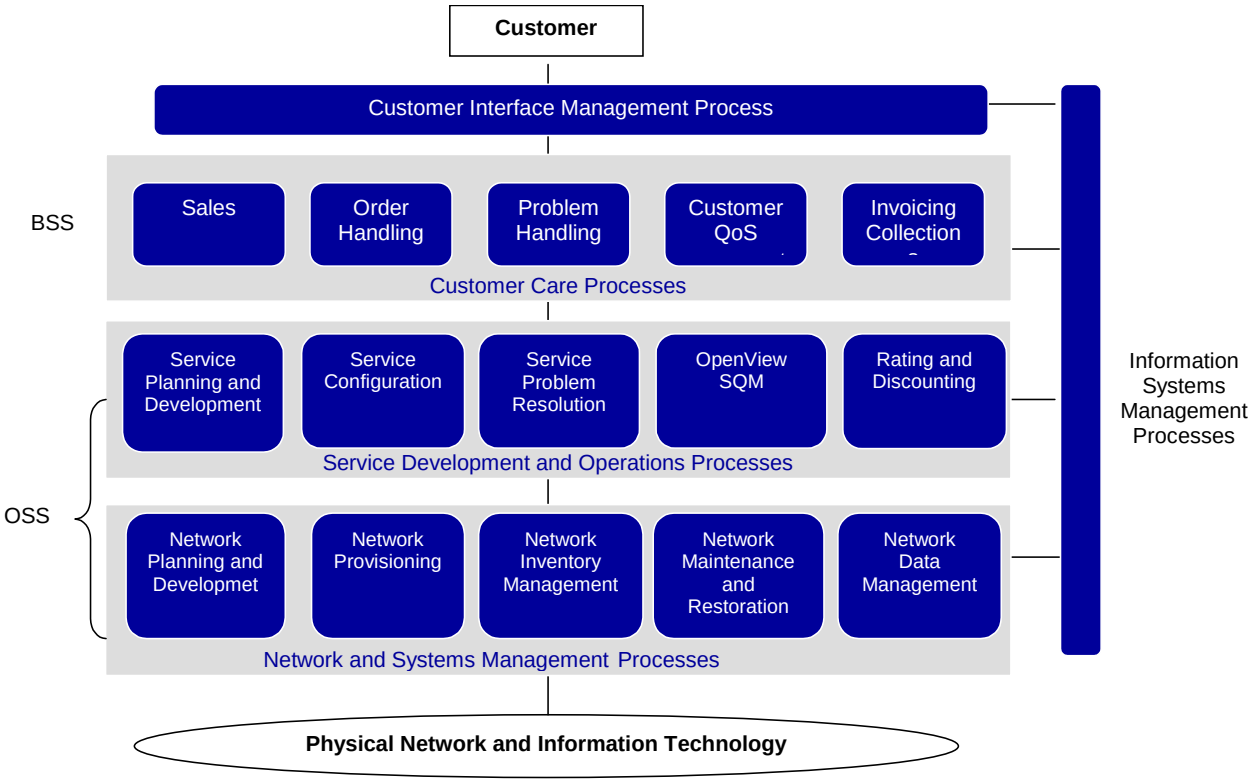
- They usually cannot process, calculate, and report on the end-to-end service metrics. Instead, they focus on a specific technology.
- Network-based performance information is not sufficient to represent customer perceived service quality.
- The evaluation of overall service quality relies on aggregating information in real-time.

Managing service quality requires a new approach based on an integrated network operation support system (OSS) layer and a new class of applications, called service level integrators.

Management of service quality occurs in the OSS stack at the service management level as shown in Figure 1.

OpenView SQM obtains an overall view of service quality by interacting with the traditional performance management systems at the network management level, as well as other service level systems and processes. For complex services, OpenView SQM computes a synthetic view of the service quality.

Figure 1 Service Quality Management



1.3.3 Benefits of Service Quality Management

Service quality management brings valuable benefits to your business. The following sections describe these benefits in detail.

1.3.3.1 Proactive Service Management

Service quality management gives service providers the information they need to manage services proactively instead of reactively. Proactive management allows you to prevent problems rather than waste time solving the problems once they have already caused damage. Proactive service management gives your business the following benefits:

- Improved service quality.

  Because the service management solution informs operators as soon as the service degrades, they can act immediately to restore the expected service quality.

- Penalities avoided.

  You have information about the service quality at all times and can prove that you are respecting your SLAs in case of misunderstandings or mistrust.

  You can closely monitor the SLA associated with especially high penalties so that SLA violations do not occur.

- Improved resource planning.

  Reports on the overall service quality to help you understand future evolution of your service and plan the infrastructure for these future needs. The reports

optimize how you use resources so you can forecast whether you need to invest in new resources.

- Service differentiation.

  You can offer attractive SLAs to differentiate your service from competitors.

- Customer loyalty and retention.

  Service quality management focuses on the service quality as perceived by the end-user, giving you a better understanding of customer expectations so that you can improve your customer relationships.

1.3.3.2 Service Level Agreement Monitoring Automation

Service quality management also automates the SLA monitoring process. Automation of SLA monitoring transfers heavy manual processes to a single system, reducing costs.

1.4 About Service Level Agreements

An SLA is a contract that specifies, among other things, service quality criteria and values against the criteria that the service provider must achieve to satisfy the customer. These criteria often relate to the quality of service. The SLA expresses criteria using service levels and thresholds.

This section describes the role of SLAs and service levels, and introduces the SLA lifecycle.

1.4.1 Role of SLAs

OpenView SQM supports two types of service level agreements:

- Customer SLAs.

  A customer SLA is a set of service quality parameters with threshold values that characterize the service quality needed for a service sold to a customer.

- Operational SLAs.

  An operational SLA is a set of service quality parameters with threshold values that characterize the quality of a service provided by a department within the service provider’s organization.

  These SLAs do not apply to a particular customer and allow you to:

  Test future customer SLAs by checking whether you can achieve future service quality goals.

  Prioritize work within the organization.

  Improve the performance of internal processes.

  Manage the relationship between the different departments.

1.4.2 About Service Levels

A service level describes a service provider's objectives for a given service. A service level gives a set of service quality parameters needed for controlling the quality of service along with their threshold values.

When selling a service to a customer, the service provider often has various marketing offers for the same service, such as a “gold”, a “silver”, and a “bronze” quality of service offer. Each of these offers corresponds to a quality of service, a price, and a specific service level.

The service quality parameters are usually specific to the service and should be easy for the customer to understand.

The difference between the “gold”, “silver”, and “bronze” service levels depends on the threshold values, which fix the agreed level of quality of service to be delivered. For example, the “gold” service level would give higher bandwidth than the “silver” and “bronze” service levels.

1.4.3 About the SLA Lifecycle

The SLA lifecycle describes the main functional components needed to manage service level and SLAs. It includes the following phases:

- **Service design phase**

  In the service design phase, you identify all the technical resources needed to provide an end-to-end service. You model each of these resources as a service component and describe the service components with a set of parameters. The service model allows you to use these parameters to evaluate the service quality you deliver.

  The result of this phase is a unique service class, broken down into service components that provide the relevant service quality parameters.

  The service design phase is a critical, stand-alone process that aims to create a common definition of services, service parameters, and service level objectives. These common definitions can then be shared between the service provider, users, suppliers, and partners.

- **Class of service definition phase**

  Once you have described a service and selected the service parameters used to evaluate the quality of service, you must define a standard class of service, or **service level**, for that service.

  A service level consists of a set of objectives defined for the service parameters. Service parameters are evaluated directly from measurements made in the service infrastructure, or computed from the evaluated parameters. Different service levels correspond to different objectives set for the same service parameters.

  Service levels do not need to be defined for a specific customer, but instead can be a part of a general service definition.

- **Service instantiation phase**

  Before the service instantiation phase, a negotiation and sales phase occurs that leads to an SLA contract signature for a particular service. Negotiation and sales is outside the scope of the SLA lifecycle.

  In the service instantiation phase, a new service needs to have its quality monitored. For example, a service model needs to be instantiated when a customer buys a new service. Service quality management begins as soon as you launch the service.

  This phase results in a service instance and triggers the collection of service quality parameters.

- **SLA creation phase**

  In this phase, the service provider creates an SLA for a specific service instance or a group of service instances, depending on the contract. The service provider also identifies the threshold values of the service level for each service quality parameter managed for the SLA.

  This phase results in an SLA that:

  Associates the service instance(s) with the service level and a customer (for customer SLAs).

Associates the service instance(s) with the service level and an internal department (for operational SLAs).

- **Service monitoring phase**

  In this phase, the service provider monitors the service and the service level agreements in real-time.

  The goal of the service monitoring phase is to avoid SLA violations by rapidly responding to service degradations.

- **Service reporting phase**

  In the service level reporting phase, the service provider generates reports about service quality.

  This phase assesses service quality on a long-term basis (monthly or quarterly) to analyze service trends or to provide customer service quality reports.

After the generation of reports, the service provider can periodically assess the service quality being provided to a particular service and the overall service quality for all customers. Depending upon new goals and changing customer needs, the service provider can make changes to the service design, completing the SLA lifecycle.

OpenView SQM provides the tools you need to easily manage each phase of the SLA lifecycle. Chapter 2 provides an overview of the OpenView SQM solution for service quality management.

Chapter 2

Overview of the Service Quality Management Solution

Chapter 1 introduced the general concepts of service quality management. This chapter describes the OpenView solution dedicated to service quality management, OpenView SQM. This chapter includes the following sections:

- Section 2.1: What is OpenView SQM?
- Section 2.2: Benefits of OpenView SQM
- Section 2.3: Example of Service Quality Management

2.1 What is OpenView SQM?

OpenView SQM provides a complete service quality management solution. It consolidates quality indicators across all domains — telecom, IT networks, servers, and applications — giving you end-to-end visibility on service quality. OpenView SQM links service quality degradations to potential affects on business, allowing you to proactively address problems and prioritize actions.

OpenView SQM monitors the service quality by aggregating information coming from all of your data sources, such as the network, the IT infrastructure, and your business processes. Using this information, you can pinpoint infrastructure problems and identify their potential affect on customers, services, and SLAs.

OpenView SQM provides a service management platform for each phase of the SLA lifecycle:

- Service design: OpenView SQM provides a vendor-neutral service object model that allows you to model any complex services.
- Service instantiation: OpenView SQM has a graphical user interface that allows service operators to define service instances. If you need an automated interface to manage service instance volumes, OpenView SQM has an open XML interface.
- SLA creation: OpenView SQM also provides tools for specifying service level agreements that are associated with service levels.
- Service level monitoring: OpenView SQM detects service quality degradations and triggers actions for SLA violations, such as alarms or emails.
- Service level reporting: OpenView SQM produces a variety of pre-defined reports on historical quality of service and statistical information. Reports can be scheduled or created on-demand. You can customize these reports to address your specific needs.

2.2 Benefits of OpenView SQM

This section describes the benefits of the OpenView SQM solution:

- Summary of OpenView SQM technical benefits.
- Integrating OpenView SQM with other environments.
- OpenView SQM availability and fault tolerance.

2.2.1 Summary of OpenView SQM Technical Benefits

OpenView SQM's key strength over the competition is that it is a total, open platform for service quality management. The main technical differentiators are:

- Open system

  OpenView SQM is a full XML engine that you can easily integrate into your OSS. OpenView SQM also has native off the shelf gateways for OpenView TeMIP, OpenView Operations, as well as an XML gateway and an SNMP gateway. OpenView SQM can naturally interface with systems dedicated to service population automation and billing adjustment, as well as CRM tools.

- Powerful service object model

  This object model allows you to monitor any complex services from the NSP, xSP, and ASP environments. OpenView SQM can supervise mobile or wireline value-added services (such as video on demand, location bases, pre-paid, and M-Commerce) and network services (UMTS, GPRS, GSM XDSL, ATM, WAP, IP-VPN).

- End-to-end service quality management view

  OpenView SQM groups together all service quality information in one system giving you a concise view of service quality across all domains.

- Follows the SLA lifecycle

  The design of OpenView SQM follows the SLA lifecycle so you can adapt quickly to rapid service evolution.

- Adaptability

  OpenView SQM can retrieve any type of service quality information to feed the parameters needed for the SLA. The OpenView SQM's service adapters allow you to connect to any system (such as network equipment, servers, databases, applications, and other OSS) through any protocol and any method (such as polling or receiving unsolicited data).

- Distribution

  You can distribute OpenView SQM, giving operators in different cities access to the system through the user interface (UI). You can also distribute the service adapters to optimize communication between the source of service quality data and OpenView SQM.

- Flexibility

  OpenView SQM is a flexible service quality management system designed to answer your needs for SLA management. You can arrange the OpenView SQM components to build a custom solution. OpenView SQM supports any source of service and service quality data.

- Scalability

  OpenView SQM consists of interoperable components that you can distribute as needed. As volumes grow, you can deploy new components to handle the new workload.

- Monitoring in real-time

  Real-time monitoring of service quality allows you to propose advanced and unique SLAs to your customers. OpenView SQM allows you to control the service quality status of services at any time as well as the compliance violation levels over a specific period.

- Full reporting

  With OpenView SQM, you can provide your customers with near real-time reports over the Web. You can schedule reports or generate them on-demand.

- Efficiency

  Service quality is managed through a single point of entry, automating service level monitoring and freeing human resources.

- High availability

  The HP-UX hardware provides high availability of the system.

- High storage capacity

  OpenView SQM relies on Oracle technology and can store a large amount of data. Moreover, with a data warehouse, critical information can be stored over the long term.

- Security

  While OpenView SQM allows you to expose some of your data to your customers, it protects the confidentiality of your data.

- Reliability

  OpenView SQM bases its fault tolerance on the Certified Message feature of the TIBCO Rendezvous software. TIBCO Rendezvous ensures message delivery and uses redundancy mechanisms to enforce fault tolerance.

2.2.2 Integrating OpenView SQM with Other Environments

This section describes how you can integrate OpenView SQM with other environments.

2.2.2.1 Off-the-Shelf XML Northbound Interface

OpenView SQM provides a plug-in in XML. You can use this plug-in to integrate OpenView SQM with external systems, such as customer relationship management, service provisioning, service ordering, billing, and existing OSS solutions.

The flexible and open architecture of OpenView SQM makes it easy to integrate it with any network, system, and application management environment, allowing you to build an integrated end-to-end service assurance solution.

The XML plug-in provides the following features:

- Billing adjustment.

  OpenView SQM provides off-the-shelf standard SLA parameters for calculating penalties, such as service availability, compliance violation levels, MTTR, and MTBF. This data is available in real-time through the user interface or through the datamart, for additional computation and integration with the billing system.

- Help desk and CRM integration.

  If a service is down or degraded, OpenView SQM can send information in real-time to the support and help desk team. This information allows the support team to repair the service and the help desk team to inform the customer of the repair. OpenView SQM, through OpenView TeMIP Fault Management, can create and

update an existing ticket or a call in a standard system, such as Peregrine ARS, Amdocs, Clarify, and EFrontOffice.

2.2.2.2 Off-the-Shelf Fault Management Integration with TeMIP

OpenView SQM provides native integration with the OpenView TeMIP Fault Management solution. OpenView SQM forwards detected service degradations and SLA violation alarms to OpenView TeMIP Fault Management. OpenView SQM also exports its service model to the OpenView TeMIP Fault Management application, allowing it to create logical map representation of the services, SLAs, and associated alarms

2.2.2.3 Off-the-Shelf message Integration with OVO

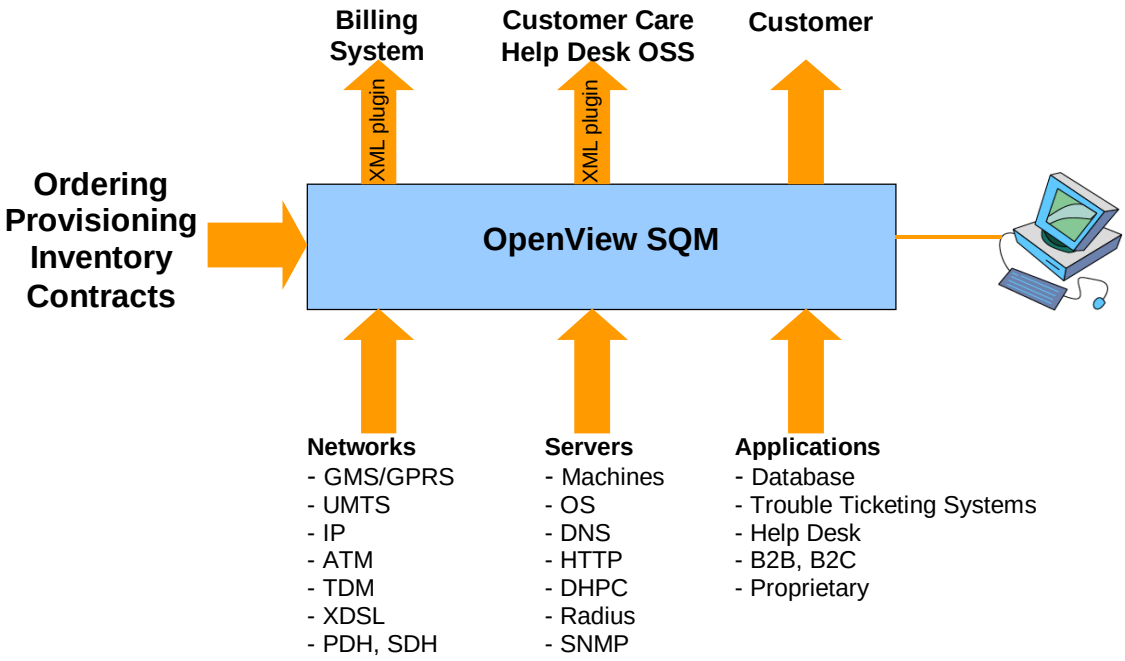
OpenView SQM provides native integration with the OpenView Operations. OpenView SQM forwards detected service degradations and SLA violation alarms to OpenView Operations.

2.2.2.4 Other Operation Support System Integration

OpenView SQM monitors services provided on a multi-technology and multi-vendor infrastructure. OpenView SQM collects information coming from networks, applications, and IT systems. This information allows service providers to assess priorities according to the affect on real-time business and decide the status of internal or customer SLAs. OpenView SQM can be deployed by integrating it into an existing OpenView TeMIP platform or OpenView Operations or as a standalone product targeted for a service operation center (SOC).

Figure 2 illustrates how OpenView SQM integrates with other operation support systems (OSS).

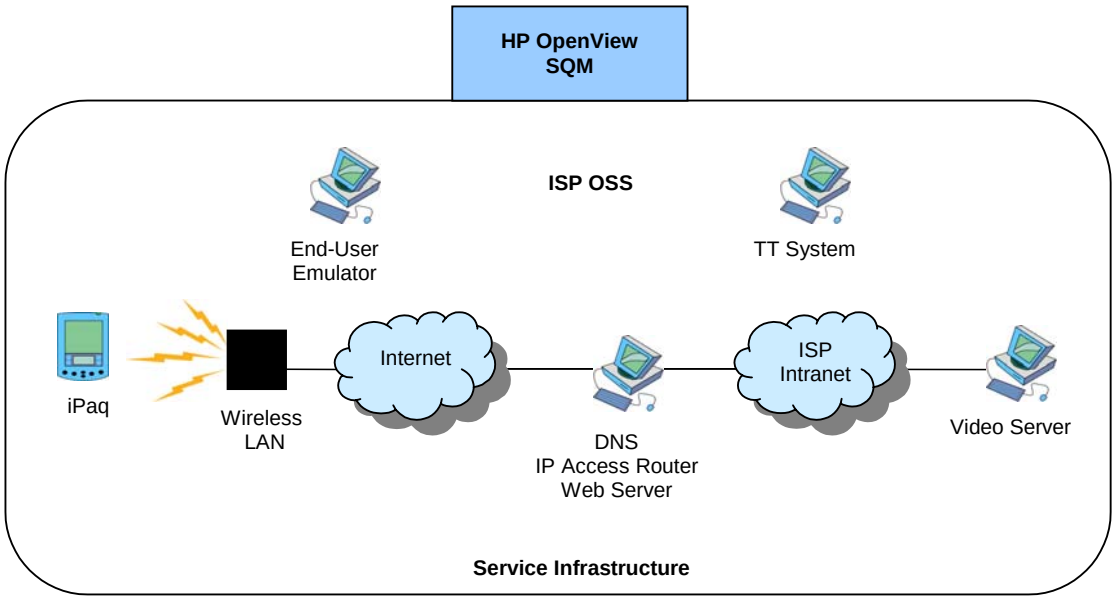
Figure 2 OpenView SQM OSS Integration



2.3 Example of Service Quality Management

This section describes an example of how OpenView SQM monitors the quality of service. An Internet service provider (ISP) wants to create a new video streaming service that allows customers to download movies over a wireless LAN and watch them on an iPaq terminal. Figure 3 illustrates the logical design of the video service.

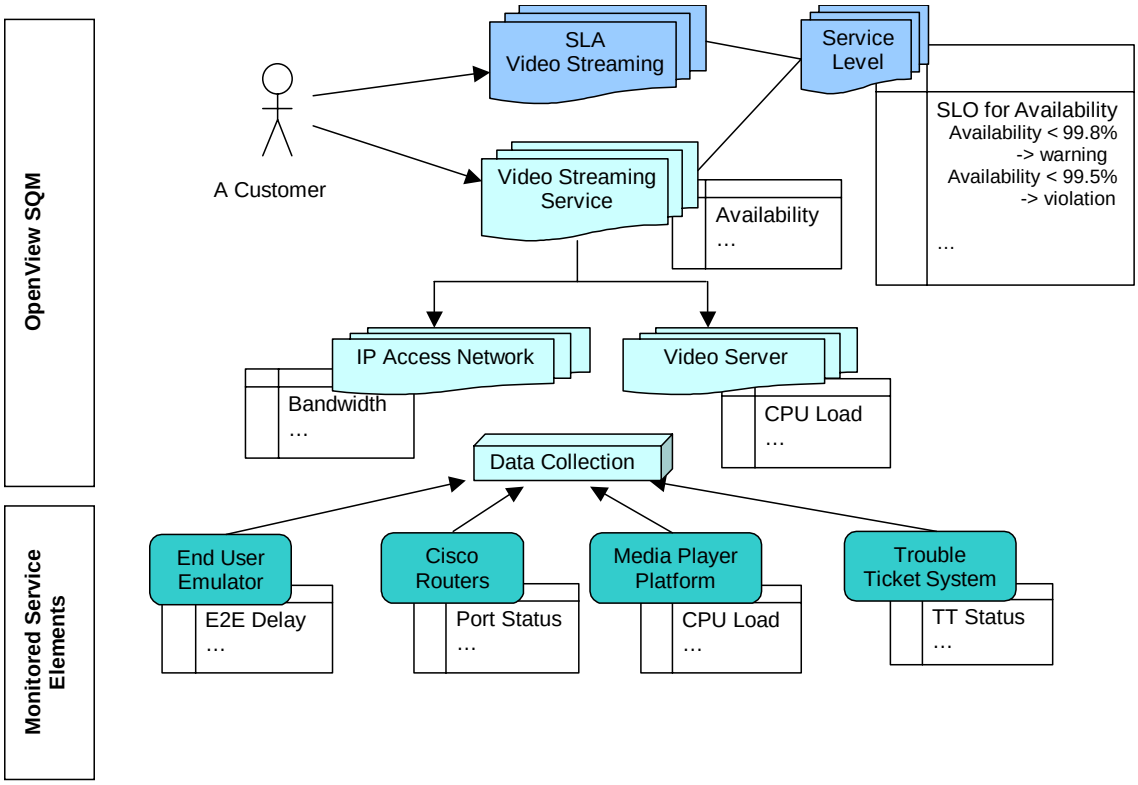
Figure 3 Video Service Design



The ISP uses OpenView SQM to monitor the quality of service agreed upon in the customer SLA. The SLA specifies several service levels associated with parameters defined in OpenView SQM, such as availability, bandwidth, and CPU load. OpenView SQM collects values for these parameters from service elements, such as the trouble ticketing (TT) system and the IP access router.

Figure 4 illustrates how OpenView SQM collects and monitors parameters of the example video service.

Figure 4 Example Service Quality Management Solution



For more details about the Video Service illustrated in Figure 4, refer to the case study in Chapter 5.

Chapter 3

Architecture of OpenView SQM

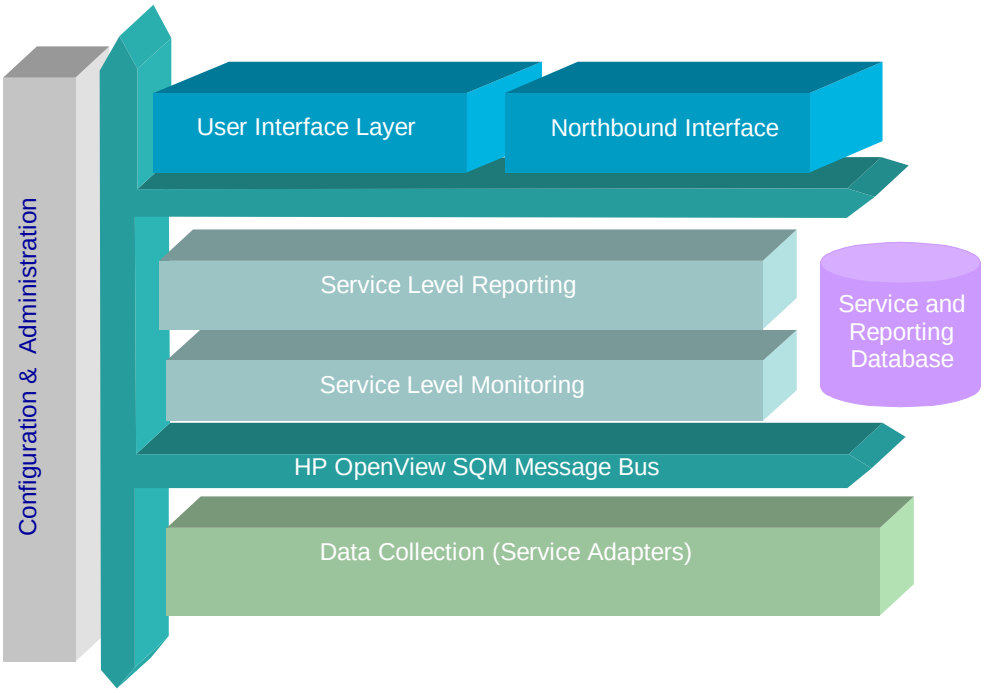
This chapter provides a detailed look at the architecture of the OpenView SQM solution. It includes the following sections:

- Section 3.1: Introduction to the OpenView SQM Component Architecture
- Section 3.2: About the Service Level Monitoring Component
- Section 3.3: About the Service Level Reporting Component
- Section 3.4: About the Data Collection Layer
- Section 3.5: About the Data Presentation Layer
- Section 3.6: About the Framework, Configuration, and Administration Components
- Section 3.7: Distributing and Scaling OpenView SQM

3.1 Introduction to the OpenView SQM Component Architecture

Figure 5 describes the architecture of the OpenView SQM components used to manage the quality of service.

Figure 5 OpenView SQM Architecture



The following sections provide details about the different components of OpenView SQM.

3.2 About the Service Level Monitoring Component

The OpenView SQM service level monitoring component is the heart of OpenView SQM. The monitoring component:

- Stores service definitions and service instances.
- Stores SLA definitions and their associated customer and service levels.
- Stores the values collected for the service quality parameters.
- Calculates and validates service quality parameter values against their associated service level objectives in real-time.
- Calculates compliance violation levels in real-time.
- Forwards any service degradations, SLA violations, and compliance violation levels to the OpenView SQM Monitoring UI.
- Executes user-defined actions in response to an SLA violation.
- Answers requests from any UI or external application.

3.3 About the Service Level Reporting Component

The service level reporting component archives service information, such as service definition updates and SLA violations, for reporting purposes.

The service level datamart used by the service level reporting layer delivers reports based on the following dimensions:

- Customer
- SLA
- Service definition
- Service instance
- Service component instance
- Time (aggregated measures, such as monthly, quarterly, and yearly)

Other reporting tools can use the datamart for reporting purposes. You can also integrate the datamart into a corporate data warehouse.

3.4 About the Data Collection Layer

The data collection layer retrieves the information needed to evaluate the service quality parameters in the SLAs. The data collection layer is composed of components called service adapters (SAs). An SA is a technology-specific plug-in that can adapt to any type of data source. The SA retrieves information either by polling or by receiving unsolicited data. You can use SAs to extract service information from the following:

- OSS:

  Customer reports (such as help desk performance)

  Radio access network (such as alarms, state, and performance statistics)

  Core network (such as alarms, state, and performance statistics)

  Value-added systems (such as IN nodes)

  Service platforms

  End-user emulators (such as network probes)

- Billing applications and customer relationship management (CRM):

  Billing systems

  Customer reports

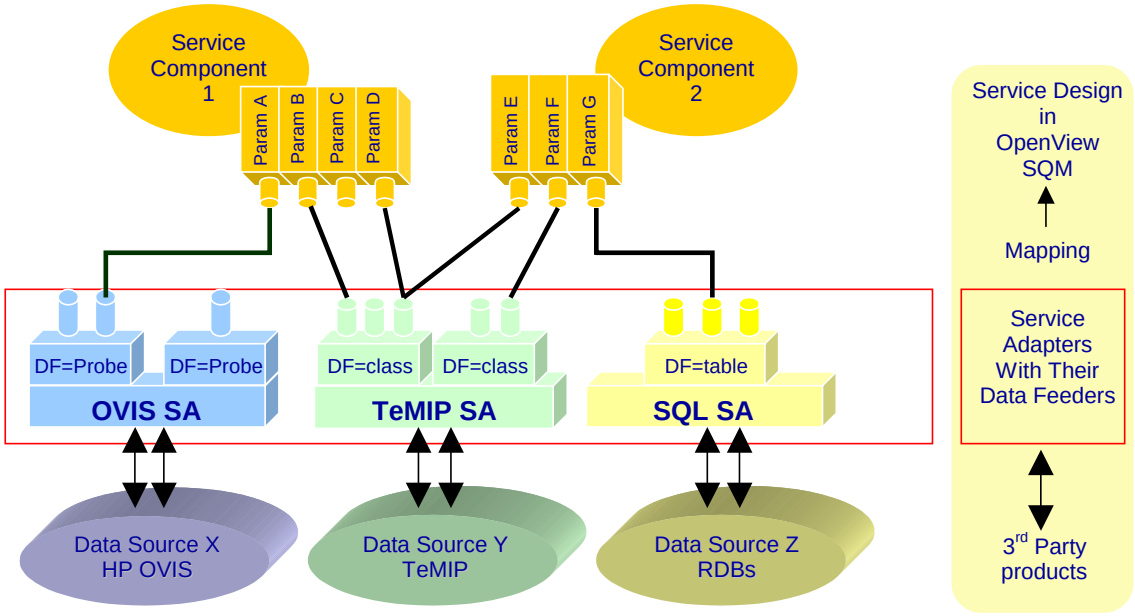
  CRM

HP proposes off-the-shelf service adapters for use by service providers. HP can provide a complete list of service adapters on request.

An SA functions as an independent application that maps the service quality data to the service quality parameters of the data feeder instance. Data feeders are OpenView SQM's source of data. For more information about data feeders, refer to section 4.3.1.2.

Figure 6 illustrates how OpenView SQM maps the parameters of the data feeder instances collected by the SAs to parameters in the OpenView SQM.

Figure 6 Service Adapter Collection Scheme



SAs also manage aspects specific to third party products, such as proprietary protocols, APIs, and databases.

3.5 About the Data Presentation Layer

The data presentation layer displays data and handles interactions between OpenView SQM and users or other OSS systems that need information for service quality management. The data presentation layer consists of graphical user interfaces and a northbound interface, components that communicates with the OSS.

The following sections describe the UIs and the northbound interfaces.

3.5.1 About the OpenView SQM User Interfaces

OpenView SQM includes a user interface for each step of the SLA lifecycle. The following sections describe each interface in detail.

3.5.1.1 OpenView Service Designer UI

The OpenView Service Designer UI allows the user to design, model, and import services in Unified Modeling Language (UML). OpenView SQM bases the OpenView Service Designer UI on the Rational Rose Modeler, software that supports the UML standard.

OpenView SQM provides predefined meta-definitions for services. You can use the OpenView Service Designer UI to customize the predefined definitions included with OpenView SQM, reducing the effort needed to design a new service. You can also create your own service definition templates that other services can use later.

In the OpenView Service Designer UI, you can fully design a service and check its model without any interaction with the other OpenView SQM components. Once you have designed and validated a service, the OpenView Service Designer UI generates an XML file representing the service. This XML file needs to be loaded in OpenView SQM.

3.5.1.2 OpenView SQM Administration UI

The OpenView SQM Administration UI is an application that allows you to:

- Create service instances using the service definitions created with the OpenView Service Designer UI. Once you enable the service, it starts collecting service quality information, validating service levels, and calculating compliance level violations.
- Create and modify service levels associated with service parameters.
- Define customer identities that will be associated with service level agreements.
- Create and modify service level agreements.
- Define thresholds and service levels.
- Check, suspend, and resume data collection by the data feeders.
- Import and export OpenView SQM objects from an XML formatted file.

You can also instantiate services through the command-line interface by importing XML files that specify service instances.

3.5.1.3 OpenView SQM Monitoring UI

The OpenView SQM Monitoring UI allows you to monitor in real-time service level agreements and their associated services. This application can work in conjunction with alarm handling systems, in particular TeMIP Alarm Handling. The SLA monitoring UI displays service violations and malfunctions in real-time. The UI provides navigation to help you understand the problem so that you can take corrective actions.

From the OpenView SQM Monitoring UI, the service operator can observe:

- The service quality measurement for any service parameter.
- The status of the measured service quality compared to the service levels required for an SLA. The status is a percentage that describes how degraded or how well a service is currently performing. Colors (red, yellow, and green) show the degree to which the service quality has degraded.
- The compliance violation level, which shows the likelihood of an SLA violation during a given period. It shows the non-compliance period observed and compares it to a maximum period authorized for non-compliance.
- The data that affects a particular SLA.

3.5.1.4 Service Level Reporting UI

The service level reporting UI allows you to build and display reports about the quality of service delivered (such as service availability and usage). The UI provides both scheduled and on-demand reports. OpenView SQM produces reports for system users or for the final customers. The Service Level Reporting UI relies on the Business Objects solution.

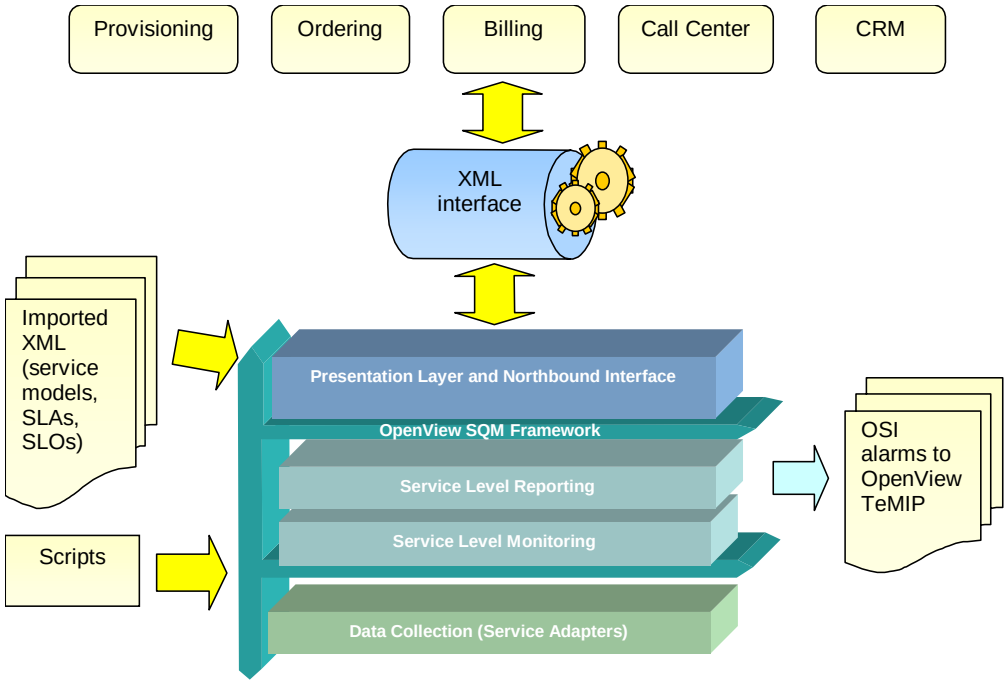
OpenView SQM provides off-the-shelf reports that are available on the Web or on Microsoft Windows. The Business Objects Broadcast Agent can produce periodic reports and send them to customers using the Infoview portal or other media, such as email, fax, and SMS. You can also use other tools to access the service level datamart and create your own custom reports.

3.5.2 About the Northbound Interfaces

Though you can deploy OpenView SQM as a stand-alone application, it is usually the main point of contact between applications, such as OSS, CRM, billing, business

intelligence (BI), provisioning, inventory, contract management, and ordering. As described in Figure 7, OpenView SQM is a complete open framework.

Figure 7 OpenView SQM Integration Capabilities



OpenView SQM gives several interfaces for integration with external applications:

- XML interface.

  OpenView SQM provides a full open XML interface based on TIBCO technology. The XML interface interfaces with business support systems (BSSs). OpenView SQM emits service degradation, SLA violations, and compliance alarms on the TIBCO bus. The external applications can use the TIBCO bus to consult, create, or update services, customers, and SLAs.

- Native integration with OpenView TeMIP Fault Management.

  OpenView SQM has an interface designed specifically for the OpenView TeMIP Fault Management solution. It exports a full OpenView TeMIP fault model in management specification language (MSL). OpenView SQM translates service degradations, SLA violations, and compliance alarms into OpenView TeMIP Fault Management OSI alarms.

  OpenView SQM has two integration points in the OpenView TeMIP OSS framework, as illustrated in Figure 8. The OpenView SQM Gateway maps OpenView SQM violation and degradation events into OSI alarms. The OpenView TeMIP Service Adapter collects data from a network managed by OpenView TeMIP Fault Management.

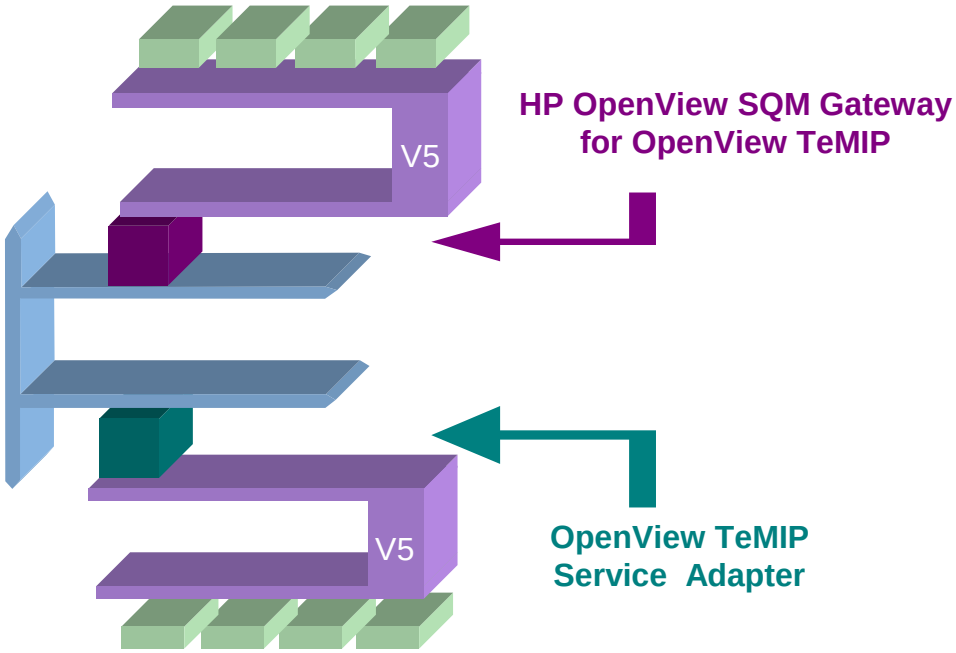
- Native integration with OpenView Operations (OVO).

  OpenView SQM has an interface designed specifically for the OpenView Operations. OpenView SQM translates service degradations, SLA violations, and compliance alarms into OpenView Operations messages.

- XML import and export.

OpenView SQM is a full XML engine. Using a command-line interface, you can import or export OpenView SQM objects (such as service models, service instances, SLAs, or service levels).

Figure 8 OpenView SQM Integration into OpenView TeMIP OSS



3.6 About the Framework, Configuration, and Administration Components

The following sections describe the framework, configuration, and administration components used by all of the other OpenView SQM components.

3.6.1 OpenView SQM Framework Components

OpenView SQM framework is based on the TIBCO Rendezvous middleware, a messaging system that enables real-time messaging for OpenView SQM applications.

You can plug any OpenView SQM application into the TIBCO Rendezvous middleware to get messages or to listen for messages exchanged with OpenView SQM.

3.6.2 Configuration and Administration Components

OpenView SQM relies on TIBCO Rendezvous tools for managing and configuring the platform. The TIBCO Rendezvous tools are responsible for the following activities:

- Centralized application configuration
- Application monitoring
- Centralized error logging

OpenView SQM provides user interfaces for each of these activities.

3.7 Distributing and Scaling OpenView SQM

The architecture of OpenView SQM supports a large number of services, SLAs, customers, and operators. You can fully distribute OpenView SQM, like the OpenView TeMIP Fault Management product.

OpenView SQM also supports vertical distribution. This vertical distribution allows you to dedicate an OpenView SQM host to one or several tiers of the OpenView SQM architecture:

- Presentation layer
- Service level reporting layer
- Service level monitoring layer
- Data collection layer

There can be several hosts of each layer type. For best performance, the service level reporting layer should run on a different host.

Chapter 4

Managing the SLA Lifecycle with OpenView SQM

The previous chapters provided a general overview of OpenView SQM and a look at the product architecture. This chapter describes how OpenView SQM manages the SLA lifecycle. This chapter includes the following sections:

- Section 4.1: Service Design
- Section 4.2: Service Level Definition Phase
- Section 4.3: Service Instantiation
- Section 4.4: Service Level Agreement Creation
- Section 4.4: Service Level Agreement Creation
- Section 4.5: Service Level Monitoring
- Section 4.6: Service Level Reporting

4.1 Service Design

During the first phase of the SLA lifecycle, a service provider designs a new service. Service design includes modeling a new service definition and identifying the sources of service quality data. The OpenView SQM object model is a powerful service object model that allows you to represent any service definition dedicated to service quality management. You can create this service definition with the OpenView Service Designer UI or through the command-line interface.

The following sections describe the OpenView SQM object model and the OpenView Service Designer UI.

4.1.1 OpenView SQM Service Object Model

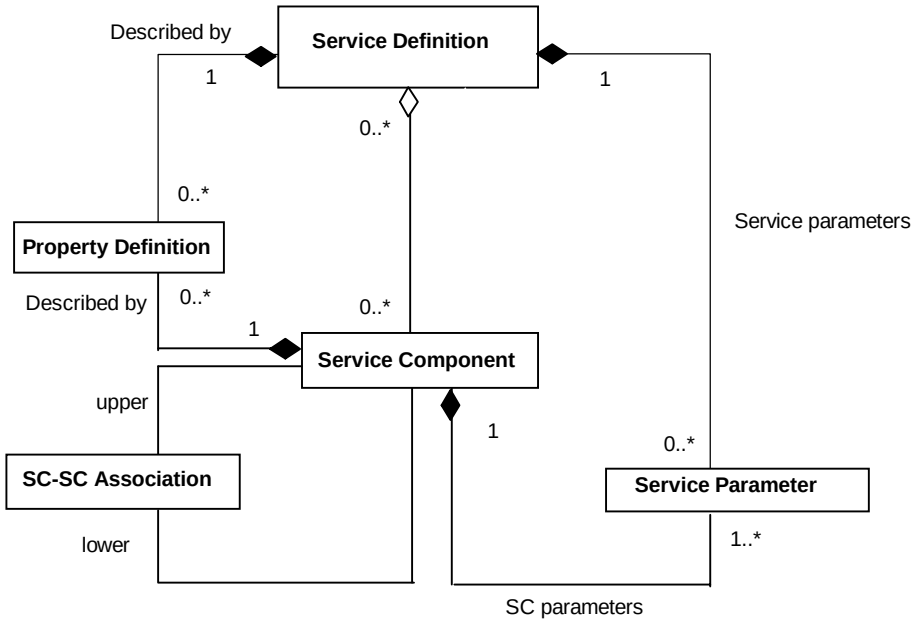
OpenView SQM uses a technology-neutral object model for modeling and defining services and sources of service quality data. You can map the object model to any service.

4.1.1.1 OpenView SQM Service Object Model Description

The OpenView SQM service object model represents a service as a collection of service components. A **service component** corresponds to hardware, software elements, or the underlying communications medium used by the service. Service and service components contain **service parameters**, values that are periodically updated and that help decide the quality of service, from either a customer or a network operator perspective.

Figure 9 illustrates in UML the service object model supported by OpenView SQM.

Figure 9 OpenView SQM Service Object Model



As shown in the Figure 9, the OpenView SQM service object model describes a generic service, which is composed of service components. These service components can themselves be composed of other service components. Both a service and a service component can have service parameters.

For more information about the UML conventions used in the figure, refer to Appendix C.

4.1.1.2 Main Features of the Service Object Model

The OpenView SQM service object model is:

- Generic.
- Technology neutral and vendor neutral.
- Designed so that you can model any complex service for service quality management.
- Designed to answer the needs of service quality management.
- Designed to give you a large degree of freedom so that you can drive how you represent services. You can add new services and service component to adapt to rapid service evolution. You can also reuse and share service components to optimize resources and deployment time.

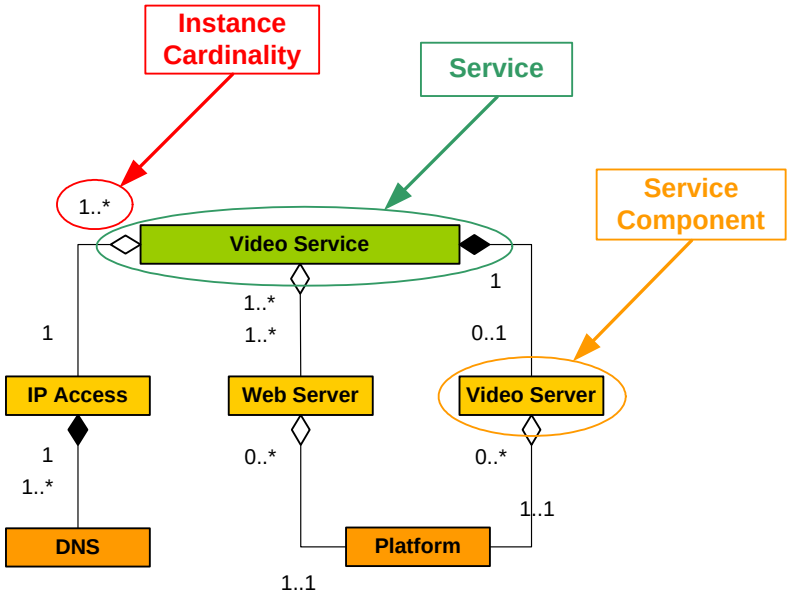
4.1.2 Creating a Custom Service Model

The service object model defines the rules to follow to design a service. The custom service model applies these rules for a specific service. The following sections provide an example of a custom service model and describe useful features of the model.

4.1.2.1 Video Service Model Example

You can use the neutral service object model defined in Figure 9 to model any service. For example, an ISP uses the service object model to model a multi-customer video service. The video service is composed of a video server and several web servers. Figure 10 illustrates the service model for the new video service.

Figure 10 Video Service Model

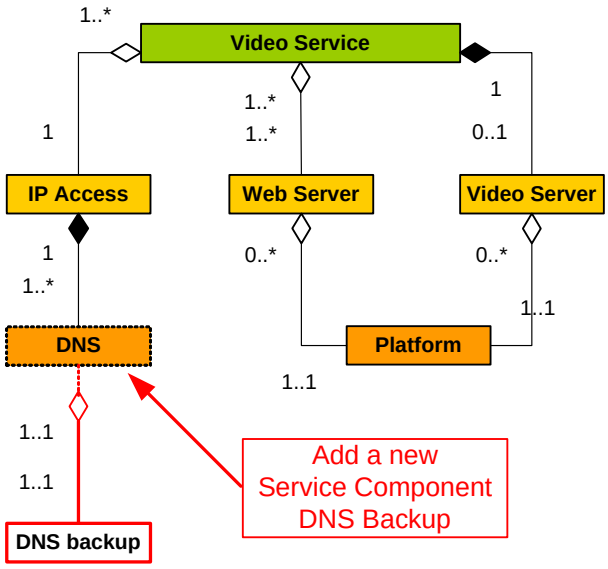


During the design phase, the ISP defines the service, service components, and service parameters for the video service using the OpenView Service Designer UI described in section 4.1.3. Later in the SLA lifecycle, the service definition can be instantiated to support customers and the SLAs associated with these customers.

4.1.2.2 Useful Features of the Custom Object Model

Your custom object model can evolve as your service evolves. For example, Figure 11 illustrates how the ISP adds a new service component, DNS backup, after creating the video service.

Figure 11 Adding a New Component



Because you define service components independently, other services can reuse them. OpenView SQM monitors these reused components separately. When you make modifications to the reused service components, OpenView SQM makes the changes automatically and transparently to both services.

For example, an ISP decides to create a mail service. This mail service reuses the IP Access component defined for the video service illustrated in Figure 10. Figure 12 illustrates how the services reuse the IP Access component.

Figure 12 Reusing a Service Component

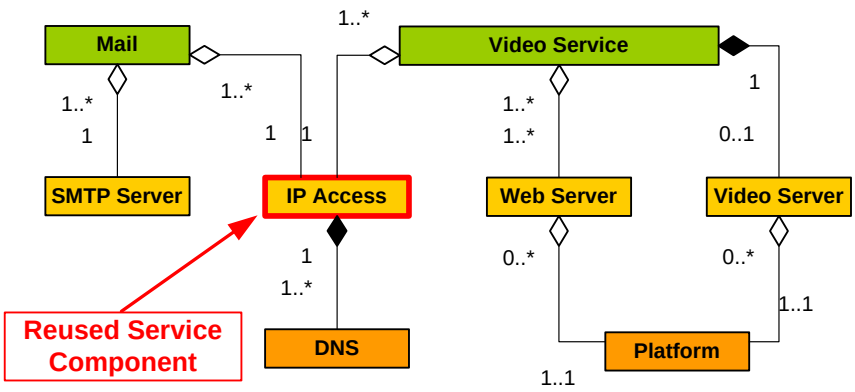
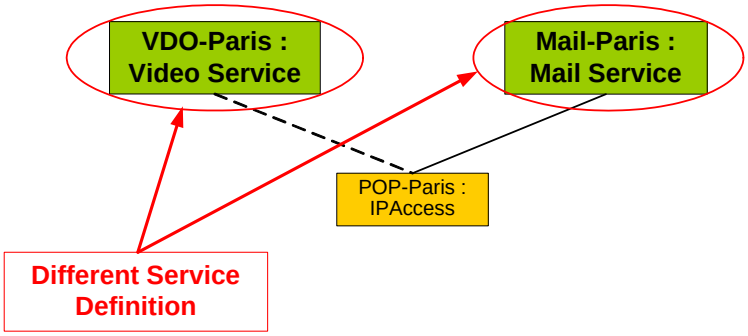


Figure 13 illustrates how different service definitions share the instance.

Figure 13 Sharing a Service Component



When the IPAccess component is loaded, OpenView SQM checks that the duplicate definition of the IPAccess component used by the mail service matches the master definition managed by the video service.

4.1.2.3 About Service Parameters

The OpenView SQM object model defines service parameters that it measures for customers. Service parameters contain quality of service metrics, state information, usage information, and characteristics.

All service parameters must specify whether their value is for a single customer (such as the number of videos sent) or whether it applies to all customers (such as the total CPU load for a host). Service parameters can be one of several primitive data types, such as strings, integers, and enumerations.

Table 1 lists several example parameters.

Table 1 Sample Parameters

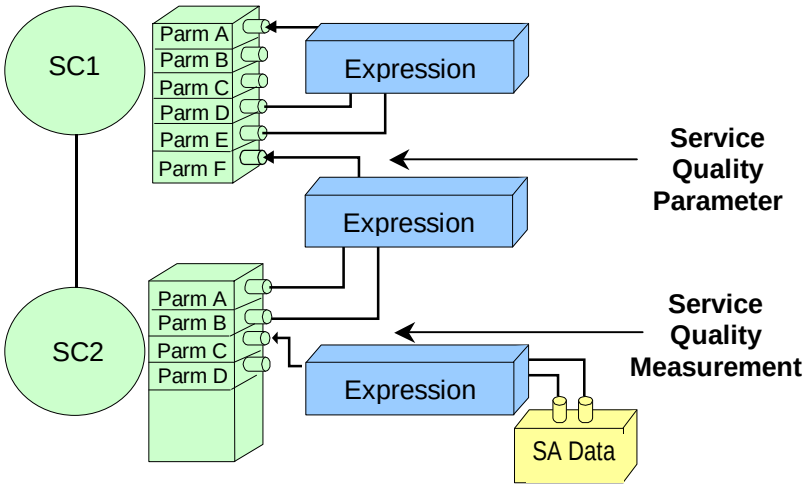
Parameter Name	Data Type
DownloadDuration	Integer
OperationalState	Enumeration
Throughput	Float

The OpenView SQM object model defines two kinds of parameters:

- Service quality measurements, which OpenView SQM computes directly from parameters collected by the service adapters. OpenView SQM includes predefined, Java expressions for computing service quality measurements (such as assign, sum, and divide). You can create your own expressions in Java.
- Service quality parameters, which OpenView SQM computes from other service parameters. OpenView SQM includes predefined, PL/SQL expressions (such as sum, average, and minimum). You can create your own expressions in PL/SQL.

Figure 14 illustrates how expressions process parameters after the service adapters have collected them.

Figure 14 Parameter Collection by Service Adapters



4.1.2.4 About Designing Data Collection

To manage the quality of service, OpenView SQM must collect information about the service components of the service. OpenView SQM uses service adapters to collect service information from the hardware and software elements supporting the service components.

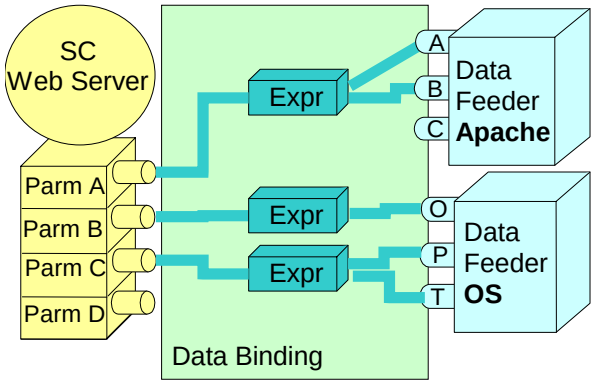
A service adapter is an independent process connected to OpenView SQM that collects data from external data sources, such as agents, probes, and monitors. It exposes one or more data feeders to OpenView SQM. Each data feeder models service resources by defining one or more service parameters. This static definition is called a **data feeder definition**.

You model service components in OpenView SQM from the top down (from the topology and customer expectations to service parameters). The data feeder parameters result from a bottom-up design, meaning that you select service parameters from existing service resources.

The binding, or association, of service quality measurements and data feeders is a part of the service definition.

Figure 15 illustrates how expressions map service parameters to data feeder parameters.

Figure 15 Parameter and Data Feeder Binding



4.1.3 Modeling and Defining Services with the OpenView Service Designer UI

The service definition is stored as an XML document. The Service Designer UI allows you to create this document through a graphical user interface. With the OpenView Service Designer UI you can:

- Model services, service components, data feeders, and their relationships with one another.
- Design expressions for parameter evaluations.
- Check model consistency.

The UI uses the Rational Rose Modeler to represent the OpenView SQM object model in Unified Modeling Language (UML). UML is an application modeling language for class and object modeling, component modeling, and distribution and deployment modeling. For more information about UML, see Appendix C.

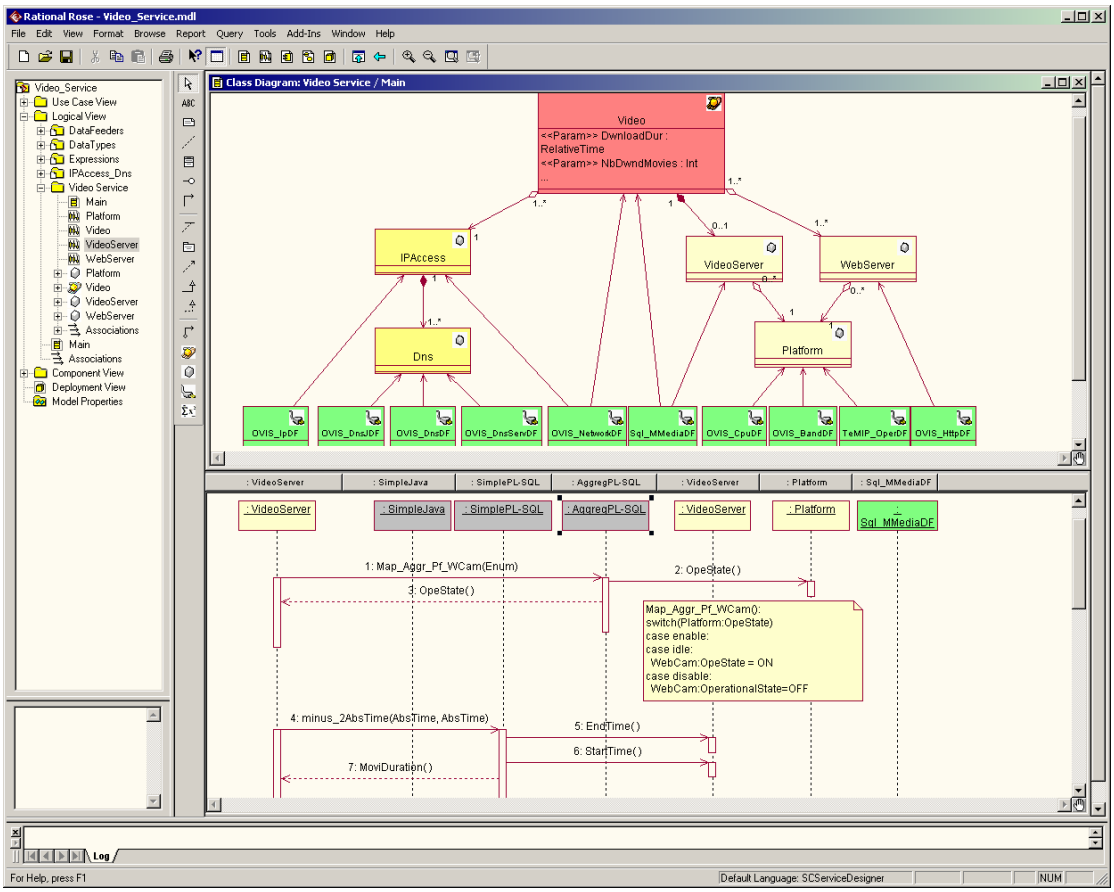
OpenView SQM provides a generic service information model that you can map to any service. The UI also provides a library of service component templates. You can reuse the service components stored in the library, allowing you to design new services quickly.

The UI represents services, service components, and data feeders as classes with relationships, such as the relationships illustrated in Figure 9. OpenView SQM gives these service elements the following class types (or “stereotypes”) in UML:

- Services have the stereotype “Service”.
- Service components have the stereotype “ServiceComponent”.
- Data feeders have the stereotype “DataFeeder”.
- Parameters of services, service components, and data feeders have the stereotype “Param”.

Figure 16 shows how the OpenView Service Designer UI models a new service.

Figure 16 OpenView Service Designer UI



As illustrated in Figure 16, the OpenView Service Designer UI represents services as UML class diagrams and expressions as UML sequence diagrams. For more information about UML diagrams, refer to Appendix C.

4.2 Service Level Definition Phase

In the service level definition phase, you define levels of service for the service and service parameters described during the service design phase. These service levels can be part of the service definition or you can link them with an SLA. For information about creating SLAs, refer to Section 4.4.

The remainder of this section describes the object model for defining service levels and how to create service levels using the OpenView SQM Administration UI.

4.2.1 About the Object Model for Defining Service Levels

In OpenView SQM, service levels are composed of one or both of the following:

- Service level objectives, which set objectives for the parameters belonging to a service. A service level objective is strictly the objective on the parameter. It can be made of one or several thresholds.
- Component service levels, which set objectives for the parameters belonging to a service component.

Each objective specifies a service degradation factor. The service degradation factor is associated with a parameter value threshold.

You can assign different objectives to the same parameter. When creating a service level, you select the objective for each parameter.

The following sections describe service level objectives, component service levels, service instance groups, and the service degradation factor in detail.

4.2.1.1 Service Level Objectives

A service level can be a collection of service level objectives. Once service parameters have been instantiated, you can add service level objectives. The service level objective is an object that allows you to define a set of parameter thresholds and give each threshold a degradation factor and an action. For example, a service level objective of "< 60%" can be added to the "CPU load" parameter.

You must always link a service level objective with a service parameter. Every time the value of the service parameter changes, OpenView SQM recomputes the service level of the service in real time.

A service level objective defines a violation threshold and can optionally define a violation clearance, one or more degradation thresholds, and a degradation clearance. You can use the following operators to define the violation or clearance level: `<`, `>`, `=`, `!=`, and `not present`.

A threshold contains a threshold level (such as reference value), a threshold type (such as degradation or violation), and a service degradation factor that varies between 0 and 100%. See section 4.2.1.3 for more information about the service degradation factor.

Each threshold can be associated with an action. By default, OpenView SQM generates an OSI alarm and sends it to the OpenView TeMIP Fault Management system.

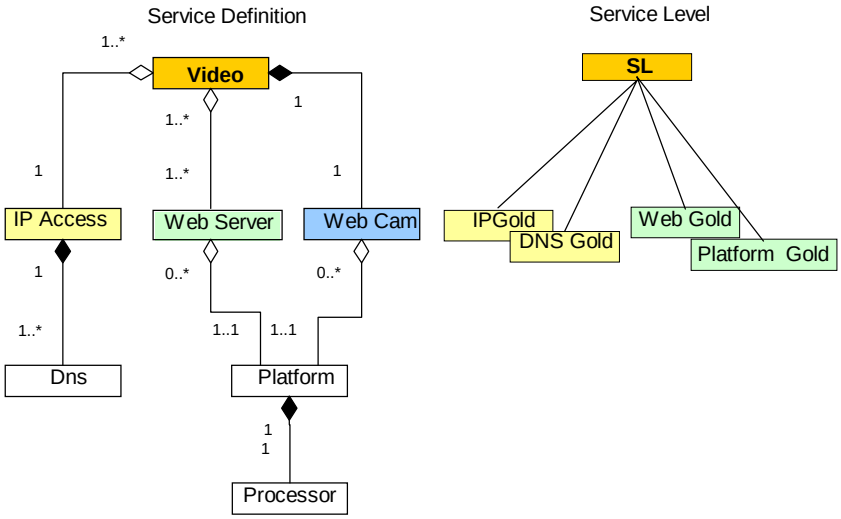
4.2.1.2 Component Service Levels

A service level can be composed of one or more component service levels. Each component service level is defined for a given service component definition.

The component service levels set service level objectives for a service component. These objectives define thresholds that OpenView SQM evaluates against the corresponding parameter values of the service component.

OpenView SQM defines component service levels under the service level in a flat model. The service level hierarchy does not include the service definition hierarchy. Figure 17 illustrates the difference between the definition and service level hierarchies.

Figure 17 Service Definition versus Service Level Hierarchy



4.2.1.3 Service Degradation Factor

The result of the validation of a parameter value against a service level objective depends on the service degradation factor. The service degradation factor varies from 0% (operational) to 100% (failure). Intermediate values characterize a degraded service.

Figure 18 illustrates several thresholds: level 0, level 1, and level 2.

Figure 18 Multiple Threshold Service Degradation

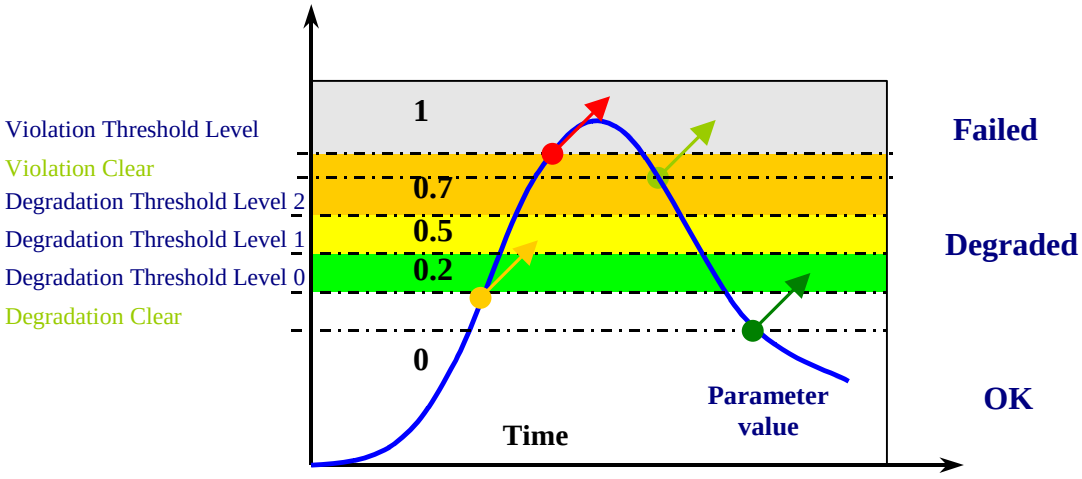


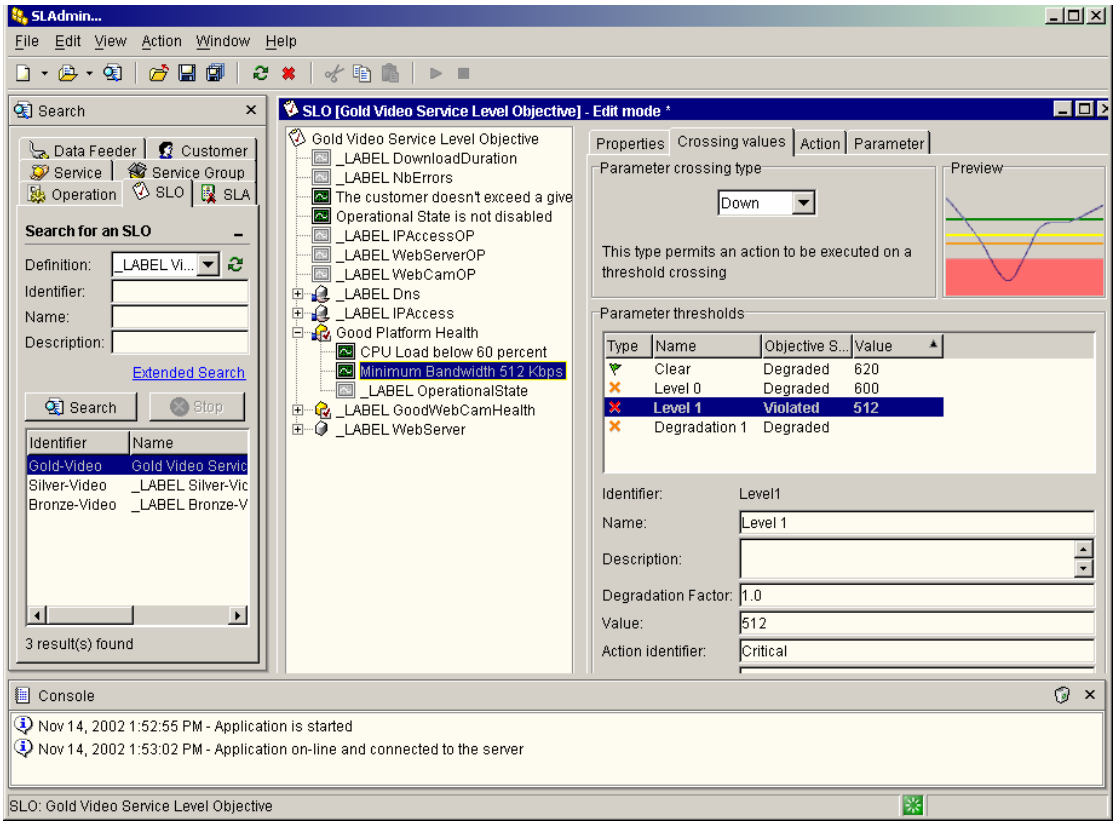
Figure 18 illustrates an objective for the CPULoad parameter. At a threshold level of zero, the CPULoad parameter has degraded by 20%. At a threshold level of one, the CPULoad has degraded by 50% and OpenView SQM issues a warning alarm. At a threshold level of two, the CPULoad has degraded by 70% and OpenView SQM sends a critical alarm, as shown by the red arrow.

4.2.2 Creating Service Levels with OpenView SQM Administration UI

You can use the OpenView SQM Administration UI to create service levels, service level objectives, the thresholds of service level objectives, and component service levels.

You create new service levels with the New Service level dialog box. Once you have created a service level by giving it a name and description, you can associate it with thresholds as illustrated in Figure 19.

Figure 19 Edit Service Level Dialog Box



4.3 Service Instantiation

Once you have designed a new service model, it must be instantiated. During instantiation, you map the definitions created during the service design phase of the SLA lifecycle to real world services, service components, and data feeder objects. Once you have deployed a service, OpenView SQM can start collecting performance information about the service and its components.

First, you deploy the service adapters. Then you create the service instances. You can create service instances manually using the Service Administration UI or automatically by exporting data from the service provisioning system.

The following sections describe how the OpenView SQM object model defines service instances and how services are instantiated using the OpenView SQM Administration UI.

4.3.1 About the Object Model for Instantiating Services

In the OpenView SQM object model, service and service component instances consist of a set of parameter instances. A data feeder or the parameters of other service components provide the value of the parameter instance.

OpenView SQM uses the values of the parameters to evaluate the quality of service delivered to the customer. During evaluation, OpenView SQM compares these values in near real-time with the objectives defined in a service level as a part of the service level agreement.

The following sections describe parameter instances, service instances, service component instances, and data feeder instances in more detail. Section 4.4 describes service level agreements and service levels.

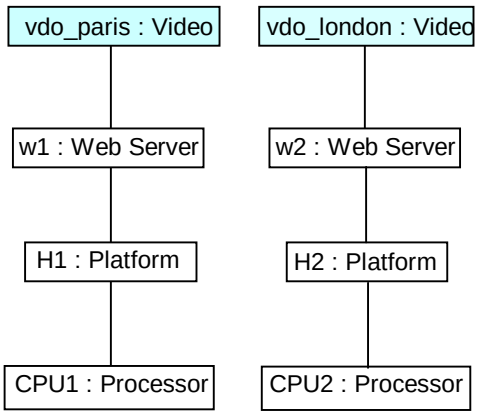
4.3.1.1 About Parameter, Service, and Service Component Instances

A parameter instance contains a value that OpenView SQM periodically updates. As described in section 4.1.2.4, OpenView SQM calculates the values of parameter instances using expressions.

A service instance is an active object built with a service definition created during the service definition phase. A complex service instance (such as a service that provide streaming video to customers in Paris) is associated with various physical elements (such as network elements, cards, and ports) and logical elements (such as paths and CPU load).

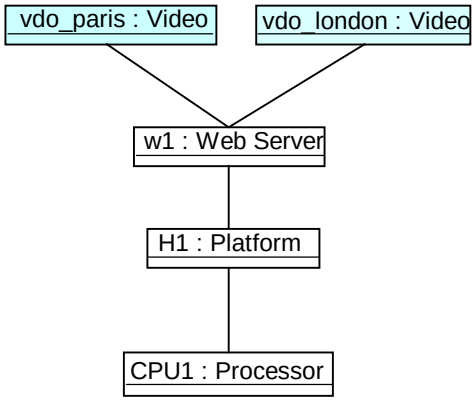
A service component instance defines a particular set of related parameters. For example, you could create an instance of a Web service component for a Web server called w1. Figure 20 illustrates an example of two video services and their respective service component instances.

Figure 20 Services and Service Component Instances



Service component instances can be shared by different service instances of the same service definition or across several different service instances of different service definitions. Figure 21 illustrates an example of a service component instance (a Web server) shared by two video service instances, Paris and London.

Figure 21 Shared Service Component Instances



You can define service and service component instances using the OpenView SQM Administration UI or the command-line interface. Section 4.3.2 describes how to create instances with the OpenView SQM Administration UI.

4.3.1.2 About Data Feeder Instances and Data Binding

A data feeder instance is a measurement point where the service adapter retrieves QoS information for a specific service. The QoS information reflects the QoS delivered and the QoS perceived by specific customers.

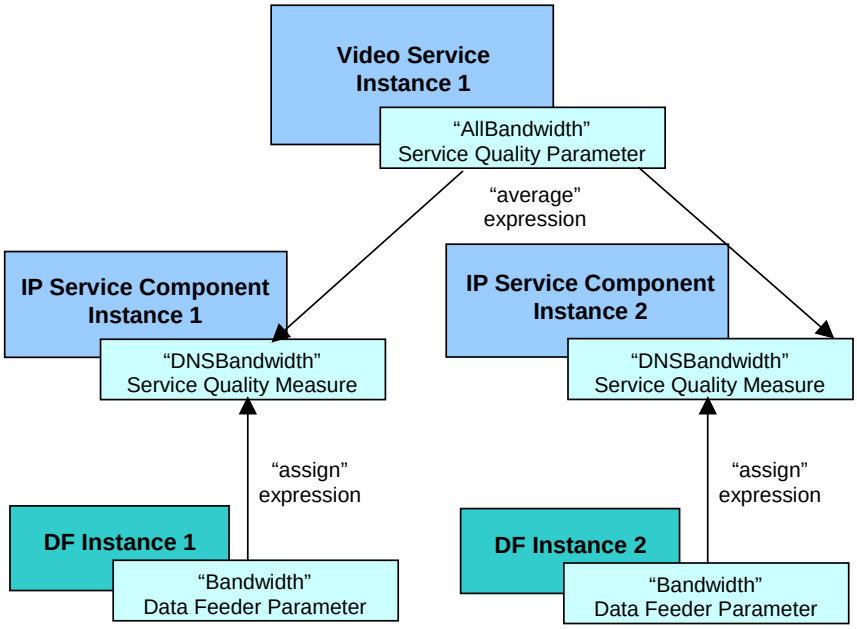
The service adapter automatically locates data feeder instances. You can also define data feeder instances with the OpenView SQM Administration UI during service instantiation. For more information about the OpenView SQM Administration UI, refer to section 4.3.2.

A data feeder instance collects parameters for a specific measurement reference point (MRP). They can collect information for three types of MRPs: direct, indirect, and contextual. The following sections describe each MRP in more detail and provide an example of data feeder data binding.

Example Data Retrieval Binding Chain

Figure 22 illustrates data feeder binding and the calculation of service quality parameters.

Figure 22 Data Feeder Binding and Service Quality Measurement



In Figure 22, a data feeder definition contains a data feeder parameter, "Bandwidth". An IP service component definition specifies a service quality measurement, "DNSBandwidth". An expression is associated with the DNSBandwidth definition. When the data feeder and IP service component are instantiated, the data feeder "Bandwidth" parameter assigns a value to the "DNSBandwidth" parameter of the IP service component.

A service definition for a video service may contain a service quality parameter, called "AllBandwidth". Because the video service can have multiple instances of the IP service component, the "AllBandwidth" parameter of the video service instance contains an average of all of the "DNSBandwidth" parameters of the IP service component instances.

4.3.2 Instantiating Services with the OpenView SQM Administration UI

You define services, service components, and data feeders in the first phase of the SLA lifecycle using the OpenView Service Designer UI. Now, you can use the OpenView SQM Administration UI to create instances of these definitions and define service levels and service level agreements to manage their quality.

Using the OpenView SQM Administration UI, you can:

- Instantiate services.
- Manage service groups.
- Manage data feeder instances.
- Manage service level agreements and service levels. Section 4.4.1 describes this aspect of the UI in detail.

OpenView SQM creates data feeder instances as persistent objects shared among multiple service components. You can also use the UI to pre-register a data feeder from a data feeder definition not defined by a service adapter.

The interface for instantiating services is an XML document. You can load this XML document in OpenView SQM through a command-line user interface.

4.4 Service Level Agreement Creation

The role of an SLA is to capture a set of service levels for a service along with details about the consequences when the service provider fails to meet the specified objectives.

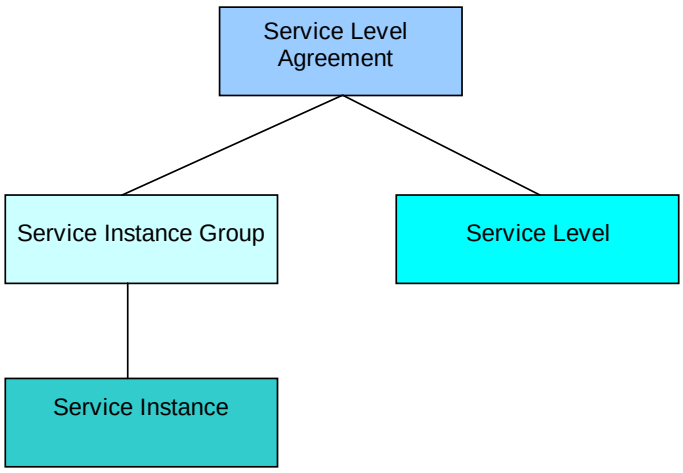
For an overview of the types of SLAs you can create, refer to section 1.4

The following sections describe the object model for setting service levels and the UI that manages service levels and SLAs.

4.4.1 About the SLA Object Model

An SLA specifies the quality of service by defining a service level for a targeted service. The customer uses a service through a set of instances, called a **service instance group**. So, an SLA consists of a service instance group that identifies the set of involved service instances and a service level that specifies the objectives. Figure 23 illustrates these parts of an SLA.

Figure 23 Parts of an SLA



For information about creating service levels, refer to Section 4.2.

4.4.2 Creating SLAs with the OpenView SQM Administration UI

The OpenView SQM Administration UI allows you to instantiate a service and to monitor its quality. With this UI, you can configure service level agreements and service instance groups.

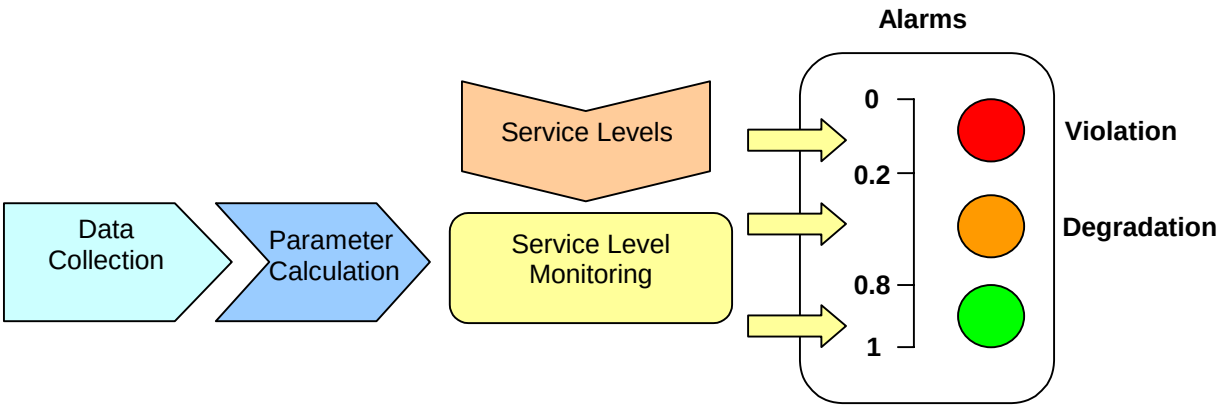
Each SLA is associated with service instance groups and service levels. You create service instance groups using the New Service Group dialog box, which allows you to give the group a name, select a service definition, and write a description. You create service levels as described in Section 4.2.2.

The interface for creating an SLA is an XML document. You can load this XML document in OpenView SQM through a command-line user interface.

4.5 Service Level Monitoring

OpenView SQM measures in real-time SLA compliance. Figure 24 illustrates how OpenView SQM monitors the quality of service.

Figure 24 Service Level Monitoring Process



OpenView SQM allows operators to:

- Monitor in real-time the service and service components in fault, any violated or degraded SLA, the affect of a service fault on the customer, and the quality of service degradation.
- Monitor the values of quality of service parameters, both in real-time and historically, including representing these values graphically in frames.

Violation detection by OpenView SQM allows operations staff to work in a proactive rather than reactive manner. The OpenView TeMIP Alarm Gateway maps service degradations and SLA violation messages to OSI alarms. The OpenView TeMIP Alarm Gateway is linked to the OpenView TeMIP Fault Management application. The OpenView TeMIP Alarm Gateway:

- Maps SLA violations and service degradations into alarms.
- Exposes the OpenView SQM service object model to the OpenView TeMIP Fault Management application. OpenView SQM uses the OpenView TeMIP Fault Management application to create service maps.
- Represents services and service components topologically, including an animated map of the quality of service state and the fault state. It generates the maps or an operator can customize and generate the maps manually.

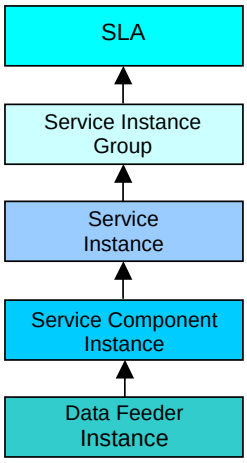
When one or several thresholds are crossed, the service level monitor notifies the appropriate action executor and gives the context of the problem. By default, the action executor sends an OSI notification using the OpenView TeMIP Alarm Gateway. You can add other actions as components that plug into the OpenView SQM bus.

For more information about the OpenView TeMIP Alarm Gateway, refer to the OpenView TeMIP OSS documentation.

4.5.1 About the Object Model for Monitoring Services

Each level of the data collection chain (data feeder instance, service instance, service instance group) implements a collection operational status and availability state. Figure 25 illustrates the object instance hierarchy.

Figure 25 Data Collection Hierarchy



The data feeder instance collects the data and propagates it to the service level agreement. Availability of information for the SLA depends on the whole instance hierarchy. For example, a binding failure between the service component instances and the data feeder instances automatically affects upper levels.

Each SLA has an administrative state that you can set to signal whether you monitor it. All instances have two state parameters:

- Operational status (disabled, enabled).
- Availability status (in test, failed, power off, off line, off duty, dependency, degraded, not installed, log full).

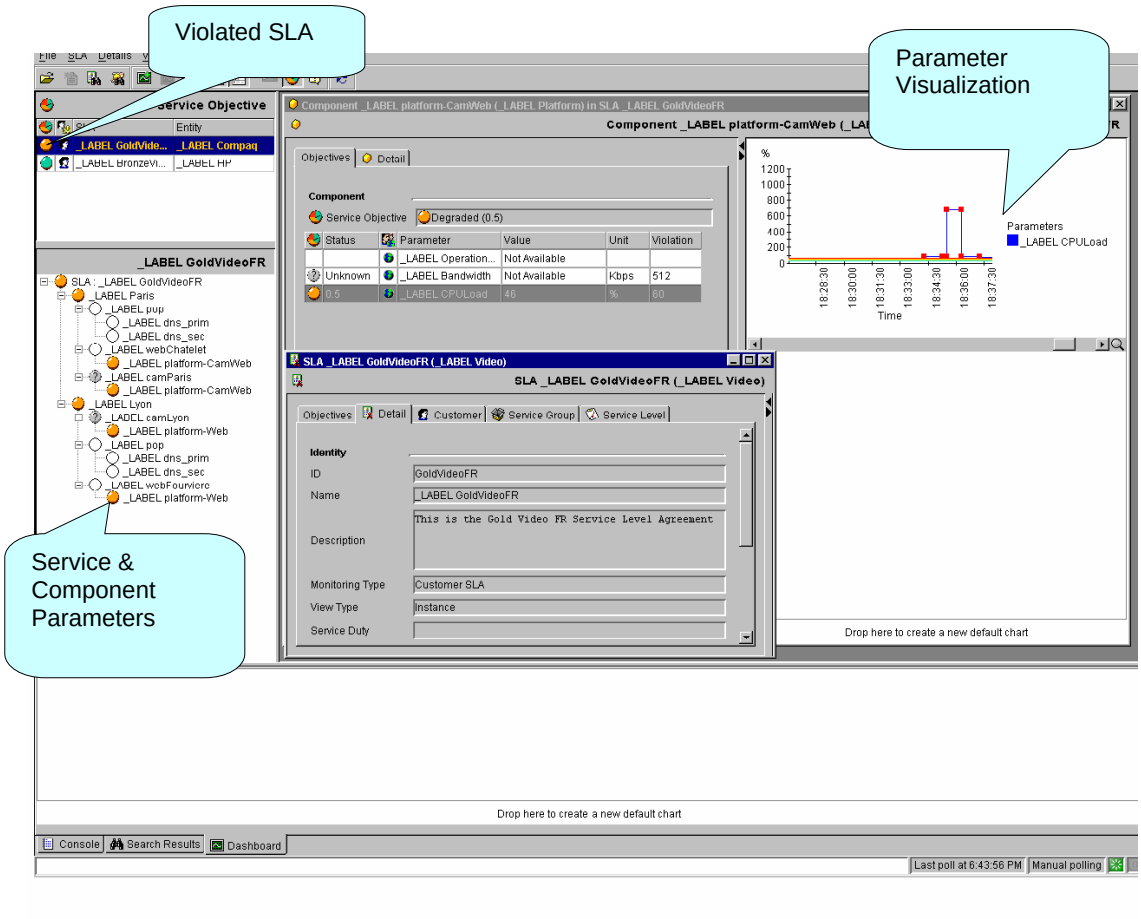
The state parameters show whether the instance is collecting all data, some data, or no data at all. OpenView SQM computes the value of this parameter from the underlying instances. So each time the collection status changes on a data feeder instance, it must recompute the collection status of all upper instances.

4.5.2 Using the OpenView SQM SLA Monitoring UI

The SLA Monitoring UI allows you to monitor in real-time the services and SLAs, from both a customer and a service provider perspective.

OpenView SQM displays service violations or malfunctions in real-time in a window that gives navigation to help you understand the problem and take corrective action. You can display service parameter values in a number of ways, including gauges and charts. Figure 26 illustrates the SLA Monitoring UI.

Figure 26 OpenView SQM SLA Monitoring UI



In addition to the features described previously, you can also use the SLA Monitoring UI to launch applications.

4.6 Service Level Reporting

OpenView SQM provides multi-dimensional service level reporting, delivering service and customer reports based on an open datamart that can be easily included in a corporate data warehouse.

This section describes the types of reports provided by OpenView SQM and the Service Level Reporting UI you can use to customize and view reports.

4.6.1 About OpenView SQM Reports

OpenView SQM produces a variety of reports on service quality, service metrics, trends, and SLA status. OpenView SQM uses Business Objects to report the service quality information.

The OpenView SQM datamart aggregates long-term indicators, such as service availability over the last week or service quality trends. The datamart stores data in an Oracle database. This database contains two types of data:

- Static data stored in static tables. The static data includes the object hierarchy, the objective status, and calculations, such as mean time between failure and mean time to repair.
- Dynamic data stored in dynamic tables. The dynamic data results from the services and service levels defined by the OpenView SQM object model.

OpenView SQM contains a collection of predefined reports, including:

- Reports on OpenView SQM data, such as service definitions, SLAs, and service levels.
- Reports representing the mean time between faults, the mean time to repair the faults, service degradation, and the availability of each service and service component.

You can do the following with the on-demand and scheduled reports:

- Display them with a graphical interface.
- Include them in the monthly customer SLA reports.
- Use them internally to assess quality degradation problems.

OpenView SQM produces the reports in HTML or PDF format, though an existing Web portal relying on a J2EE architecture can reuse the reports. You can also produce adhoc reports from the information stored in the OpenView SQM datamart.

4.6.2 Using the Service Level Reporting UI

The Service Level Reporting UI allows you to generate reports on service quality and metrics, trends, and SLA status. Business Objects provides this user interface.

You can open the Service Level Reporting UI from a Web browser because of the Web portal access provided by Business Objects and Web Intelligence reports. When you access the Service Level Reporting server through a Web browser, you must provide a login name and password.

The Service Level Reporting UI comes with an off-the-shelf reporting environment that you can use to create and display reports on static information available in the datamart. This environment contains the following reports:

- Inventory reports that provide inventory information about the service provider, including general inventory, SLA inventory, customer inventory, service inventory, and service level inventory.
- Customer inventory notice board, a report that focuses on the inventory of a particular customer.
- Service violation and degradation event reports, which include information about the total number of violations, degradations, SLA degradations, and SLA violations that affect a customer.
- Service availability reports, such as reports about the SLA service availability and service instance availability.
- Performance reports, which help provide information about the component responsible for bad service performance.

Table 2 describes the predefined reports available from the contextual menus of the Service Level Reporting UI.

Table 2 Predefined Reports

Component Type	Report
SLA	Customer degradation and violation events.
	SLA degradation and violation events.
	SLA service availability reports.
	Customer SLA service availability reports and evolution chart.
Service Instance	Service instance degradation and violation events.
	Customer aggregated view degradation and violation events.
	Service instance availability for an SLA.
	Customer aggregated view service availability (for a service instance or a service component instance).
Service Component Instance	Service component instance availability (for an SLA).
	Customer aggregated view service availability (for a service instance or a service component instance).

You can create additional reports by extending the environment and reports with the Business Object Designer and Reporter. For more information, refer to the Business Objects product documentation.

Chapter 5

Case Studies

The previous chapters described the design, architecture, and features of OpenView SQM. This chapter provides a real-world example of using OpenView SQM to design and instantiate a new service, create service levels and service level agreements, and finally implement and manage the new service and SLAs.

5.1 Video Service Case Study

This case study builds on the example introduced in Chapter 1. An Internet service provider (ISP) wants to create a new video service that allows subscribers in the United States and Europe to download movies over a wireless LAN and watch them on an iPaq terminal. The service provider offers three different levels of service (gold, silver, and bronze), each priced according to the quality of service provided.

Note that generally the design phase begins by first determining the service definition and key elements (from the top down) and then determining the different third party products that will feed the different key elements (from the bottom up). For this use case, we assume that the service provider has already carried out these steps.

This example describes the following steps in the design and management of the new value-added service:

- Modeling the video service that OpenView SQM will manage.
- Defining the video service object and its component objects.
- Instantiating the video service object and its component objects.
- Defining and instantiating the data collection objects.
- Defining service level agreements.
- Monitoring the quality of service.

The following sections describe each step of the process.

5.1.1 Modeling the Video Service

During the modeling phase, the ISP identifies the network configuration needed to support customer needs, defines how the OpenView SQM software collects data from the network hardware and software, and finally models how the components of the service interact with one another.

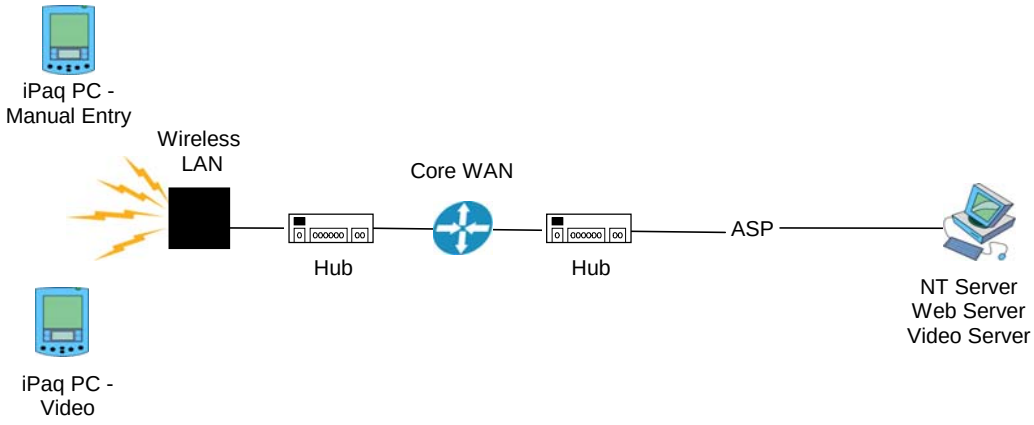
5.1.1.1 Configuring the Physical Network

The video service allows clients to watch videos using their iPaq PC. The physical architecture of the network must provide the servers and networks for sending streaming information.

For example, to support streaming video, the network infrastructure needs to provide a video server. The images pass over a wide-area network (WAN), which transforms the video protocol into a format supported by an iPaq.

The ISP creates a network to support the video service as shown in Figure 27.

Figure 27 Video Service Physical Architecture

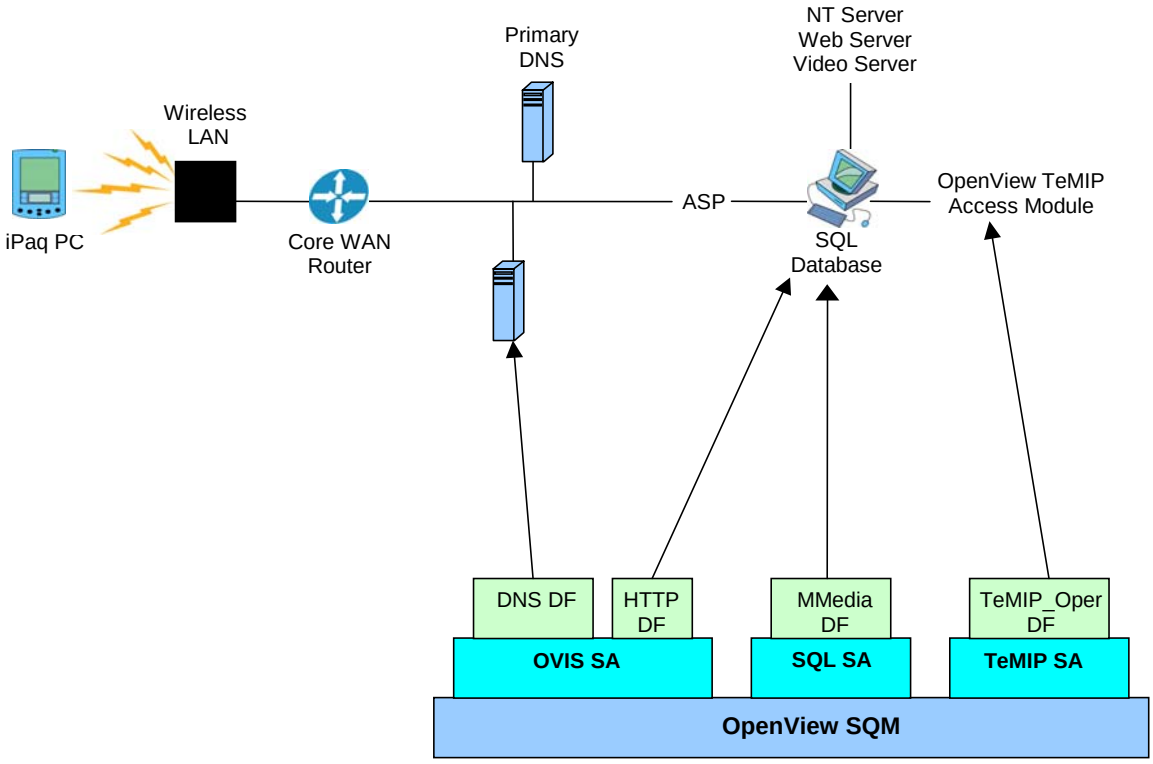


5.1.1.2 Designing the Logical Architecture

The logical architecture describes how the OpenView SQM software interacts with the ISP's network hardware to monitor the quality of the video service. OpenView SQM collects data from the network hardware and software using service adapters. A data feeder defines each measure made by the service adapter. For more information about service adapters and data feeders, see section 3.4.

Figure 28 describes where the service adapters of OpenView SQM connect to the physical network to collect data.

Figure 28 Video Service Architecture



As described in Figure 28, interactions occur through the following service adapters:

- The OVIS service adapter, which collects information from the Web server and the DNS server, such as DNS look-up time and HTTP response time.
- The SQL service adapter, which collects information from the SQL database, such as the number of downloaded movies.
- The OpenView TeMIP service adapter, which collects information from the server platforms, such as the platform operational state.

Section 5.1.4 describes how to define and instantiate the data feeder objects of each service adapter.

5.1.1.3 Designing the Video Service Object Model

Having identified the physical and logical architectures of the new service, the service provider can now model the new service.

The video service object consists of five service component objects:

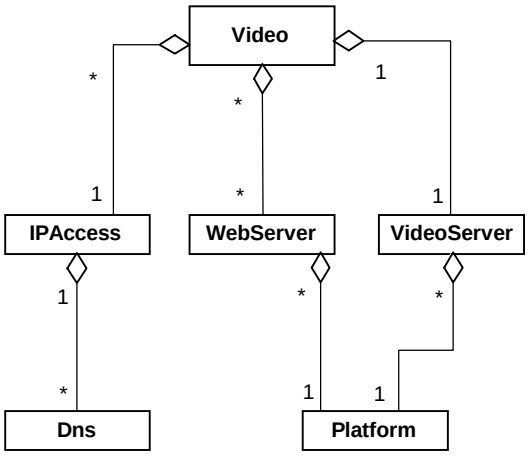
- IPAccess component object.
- WebServer component object.
- VideoServer component object.
- DNS component object.

- Platform component object, which the WebServer and VideoServer objects share.

You can add other services, such as a mail service object, that share components with the video service object. For example, a new mail service object could reuse the IPAccess component object of the video service.

Figure 29 illustrates the UML object model for the video service.

Figure 29 Video Service Object Model

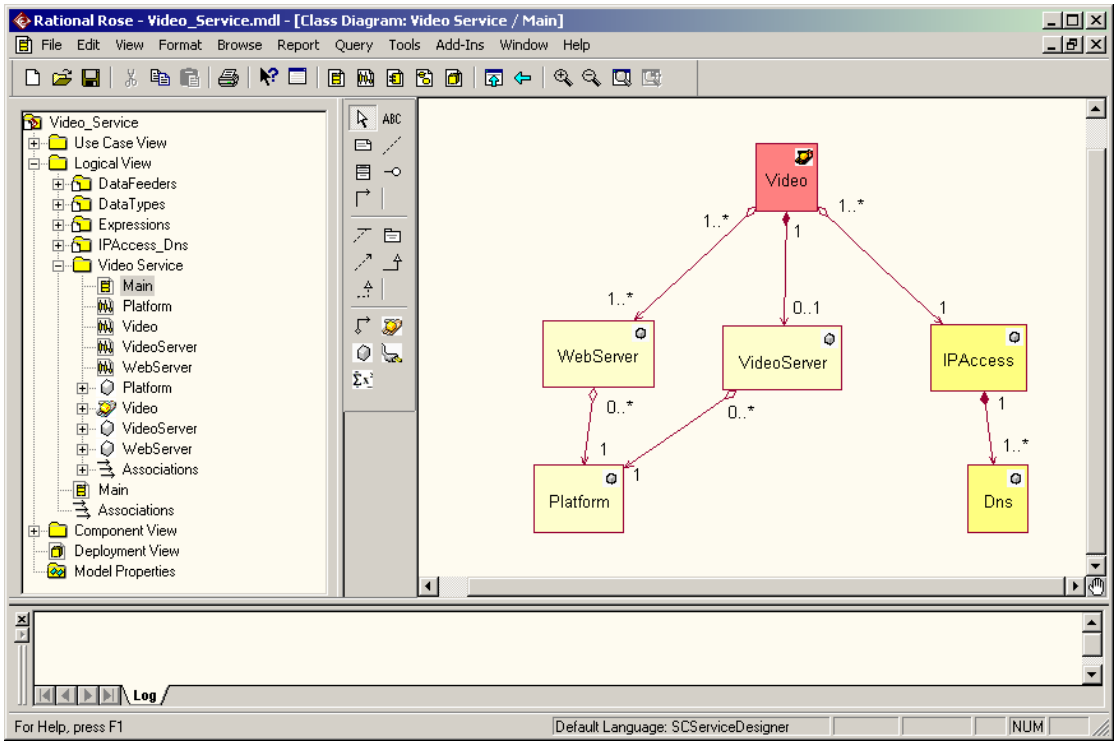


The object diagram describes the relationship between the video service object and its component objects. The numbers in the diagram give the **cardinality**, or how many instances of each component object can be present.

For example, the cardinality of the Platform object (1) indicates that a WebServer or a VideoServer runs on one and only one platform.

In the OpenView Service Designer UI, the object model of the video service appears as illustrated in Figure 30.

Figure 30 OpenView Service Designer UI: Video Service Object Model



5.1.2 Defining the Video Service Object and Its Component Objects

After identifying the video service object and service component objects needed for the new service, the ISP defines the objects. Once defined, the administrator creates specific instances of these objects to support the logical architecture produced by the design phase.

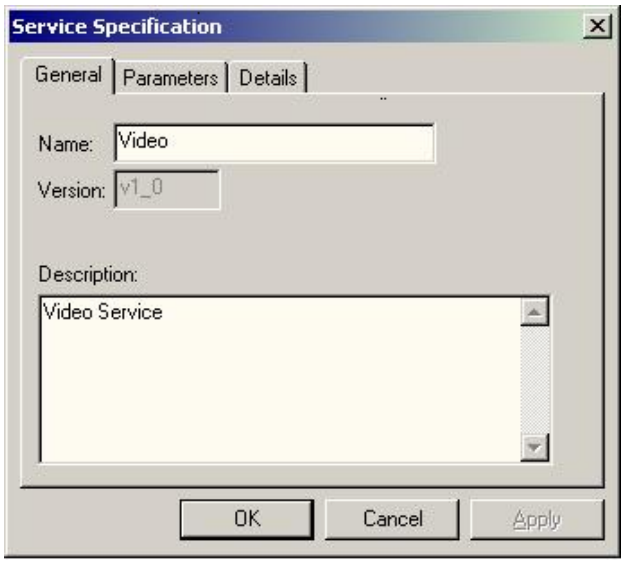
This section describes how to define the video service object and each of its component objects. Section 5.1.3 describes how to instantiate the video service object and its component objects.

5.1.2.1 Defining Video Service Object

An administrator at the ISP creates a definition of the video service object using the OpenView Service Designer UI. This service definition defines the relationship between the service object, its service component objects, and their associated parameters.

The administrator gives the service a description as described in Figure 31.

Figure 31 OpenView Service Designer UI: Defining Video Service Object

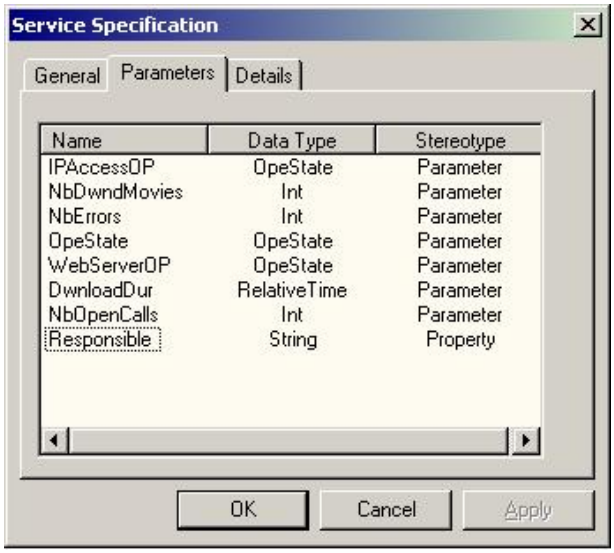


Next, the administrator gives the service a set of parameters that OpenView SQM collects, aggregates, and propagates after service deployment. OpenView SQM checks these parameters against the objectives for the customers. If degraded and violated, it triggers alarms.

The administrator defines a video service object using parameters and properties ("static" parameters). For example, the NbDwndMovies parameter specifies the total number of downloaded movies.

Figure 32 illustrates how parameters and properties are set using the Attributes tab in the OpenView Service Designer UI.

Figure 32 OpenView Service Designer UI: Specifying Video Service Parameters



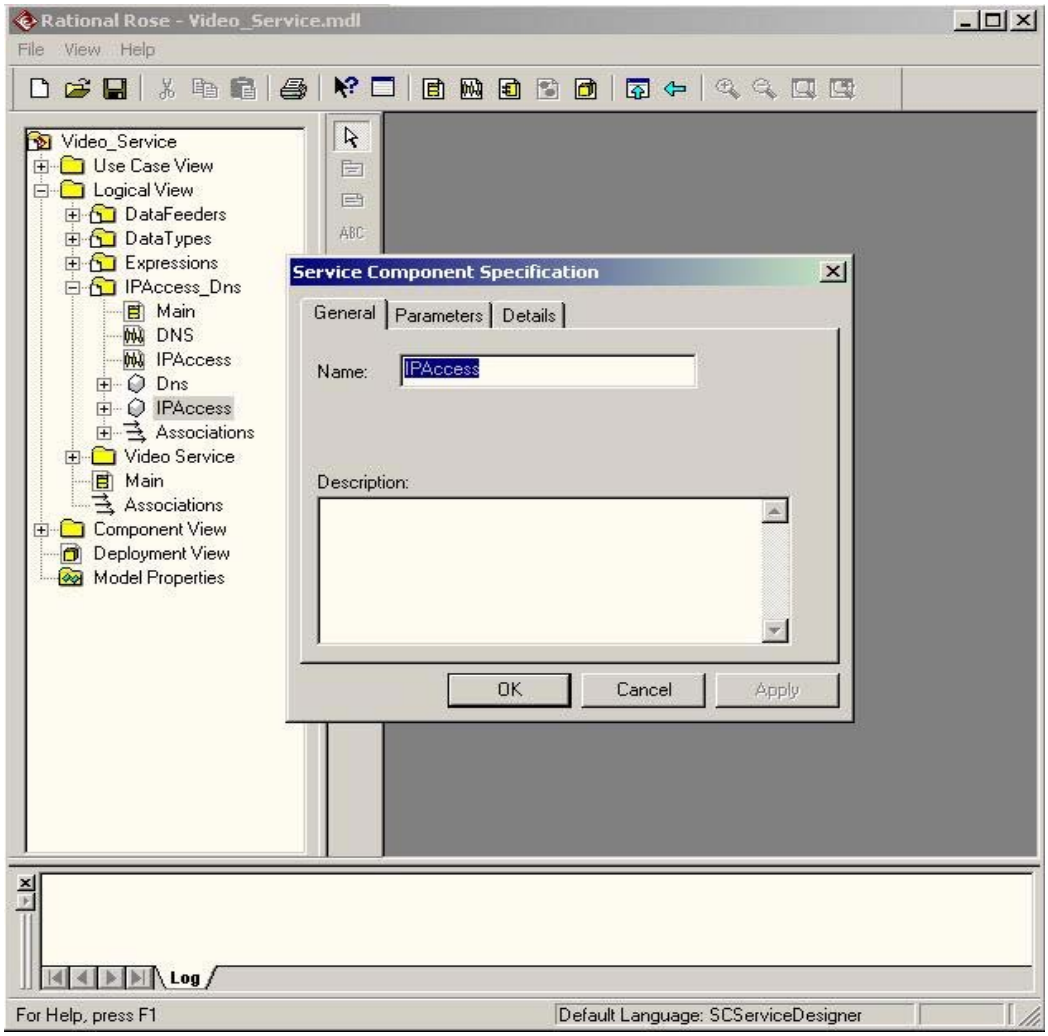
Each service parameter has an associated set of properties and values. For example, the NbDwndMovies parameter has the following properties and values:

- A data type.
- Customer dependence.
- A primary/secondary flag that shows if the parameter is primary (service quality measurement) or secondary (service quality parameter).
- An expression.

5.1.2.2 Defining the IPAccess Component Object

The administrator defines the IPAccess component object in the Service Definition UI by creating a new class as described in Figure 33.

Figure 33 Defining the IPAccess Component Object in the Service



The following user-defined parameters define the IPAccess component object:

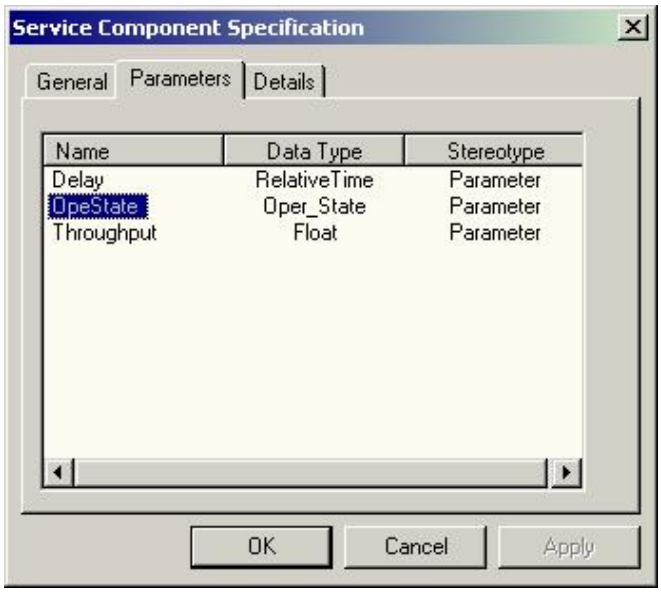
- OperationalState: the operational state of the IP Access component.
- Throughput: the connection throughput.
- Delay: the DNS name look-up delay.

The Delay parameter is a service quality parameter, meaning that OpenView SQM calculates it from one or more properties of other services. The administrator uses the Average predefined expression that assigns the average NameLookupDelay of all the DNS component instances to the Delay parameter.

These parameters are set using the Attributes tab as illustrated in

Figure **34**.

Figure 34 Setting the Parameters of the IPAccess Component Object



5.1.2.3 Defining the WebServer Component Object

The administrator defines the WebServer component object in the Service Definition UI. The WebServer component object has the following user-defined parameters:

- NbHits: the number of hits on the server.
- ResponseTime: the time it takes the Web server to respond.
- OperationalState: the operational state of the IP Access component.

The OperationalState parameter is a service quality parameter. The administrator uses the Assign predefined expression to assign the operational state of the corresponding platform to the OperationalState parameter.

5.1.2.4 Defining the VideoServer Component Object

The VideoServer component object is defined in the Service Definition UI as described for the previous service component objects.

The VideoServer component object has the following user-defined parameters:

- OperationalState: the operational state of the video server.
- MovieDuration: the duration of the movie.
- StartTime: the time the movie started.
- EndTime: the time the movie ended.

The MovieDuration parameter is a service quality parameter. Its value is the result of the following expression: EndTime-StartTime.

5.1.2.5 Defining the Platform Component Object

The Platform component object is defined in the Service Definition UI as described for the previous service component objects.

The Platform component object has the following user-defined parameters:

- OperationalState: the operational state of the platform.

- CPULoad: the workload of the platform.
- Bandwidth: the platform's bandwidth.

5.1.2.6 Defining the DNS Component Object

The DNS component object is defined in the Service Definition UI as described for the previous service component objects.

The DNS component object has the following user-defined parameters:

- OperationalState: the DNS operational state.
- ReceivedKB: the number of kilobytes of data received on the DNS.
- SentKB: the number of kilobytes of data sent on the DNS.
- NameLookupDelay: the delay during name look-up on the DNS.
- ExchangedKB: the total number of kilobytes exchanged on the DNS.

The ExchangedKB parameter is a service quality parameter. It contains the result of the following expression: ReceivedKB + SentKB.

5.1.3 Instantiating the Video Service Object and Its Component Objects

Once the administrator has defined the service object and its component objects, he or she can create individual instances of the objects that meet the needs of the ISP.

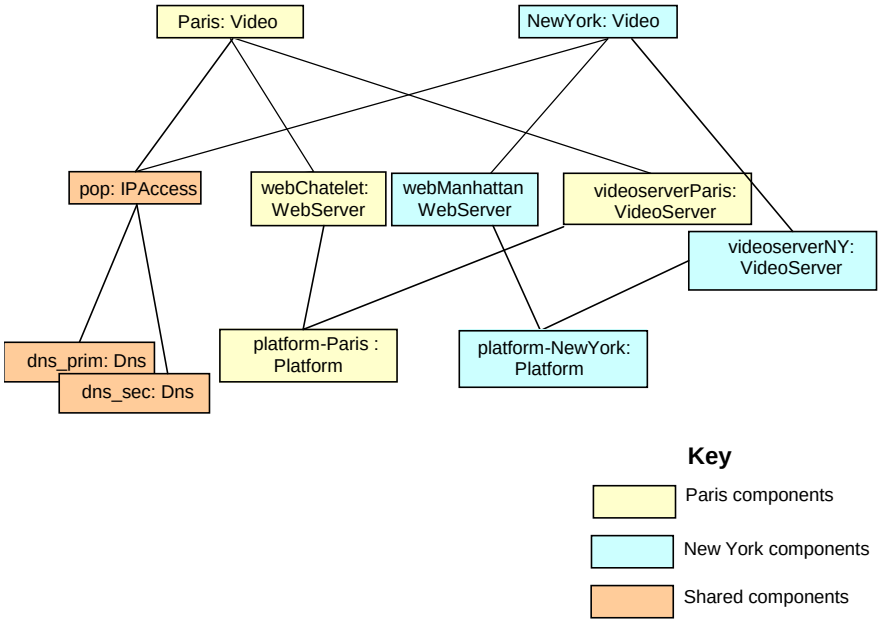
The administrator creates object instances using the OpenView SQM Administration UI. This UI selects a service definition and creates a new service instance for that definition. The administrator configures the service instance using a list of properties and user-defined values.

In this section, the ISP creates the instances of the service object and service component objects needed to support their planned video service deployment.

5.1.3.1 Overview of the Video Service Object Instantiation

The object model of the video service defined in Section 5.1.1.3 is instantiated as described in Figure 35.

Figure 35 Instantiation of the Video Service



The administrator creates two instances of the service, one for Paris and one for New York. The administrator also creates multiple instances of the service components to support each service instance.

The Paris instance of the video service has the following service component instances:

- A WebServer service component instance named webChatelet.
- A VideoServer service component instanced named videoserverParis.
- An IPAccess service component instance it shares with the New York instance of the Video service named pop.
- Two instances of the DNS service component that it shares with the New York instance of the Video service: dns_prim and dns_sec.
- A Platform service component instance named platform-Paris.

The New York instance of the video service contains the following service component instances:

- A WebServer service component instance named webManhattan.
- A VideoServer service component instanced named videoserverNY.
- An IPAccess service component instance it shares with the Paris service named pop.
- Two instances of the DNS service component that is shares with the Paris instance of the Video service: dns_prim and dns_sec.
- A Platform service component instance named platform-NewYork.

The following sections describe in detail how to create the instances of the video service object and service component objects.

5.1.3.2 Instantiating the Video Service Object

As described above, an administrator creates two instances of the video service, the Paris video service instance and the New York video service instance. The properties of each service instance are set. Table 3 lists the user-defined properties of the video service.

Table 3 Video Service Properties

Service Property	Description
Location	Provides the location of the video service.
Responsible	Provides the name of the administrator responsible for the video service.

5.1.3.3 Instantiating the Service Component Objects

The administrator creates instances of the service component objects for each video service instance. Table 4 describes the service component instances created for the video service instances.

Table 4 Service Component Object Instances

Service Component Object	Service Component Object Instances
WebServer	webChatelet webManhattan
VideoServer	videoserverParis videoserverNY
Dns	dns_prim dns_sec
Platform	platform-Paris platform-NewYork
IPAccess	pop

5.1.4 Defining and Instantiating the Data Collection Objects

The Internet service provider wants to collect the following types of data:

- Data about its customers for managing the service level agreements.
- Data about its own internal resources for proper operations and network performance as a whole.

Service adapters act as a mediation layer between the data sources and OpenView SQM. The service adapter module uses instances of the data feeder objects to collect data from one or more service components of the service.

This section describes the steps for defining and instantiating the OpenView SQM components dedicated to data collection. It describes how the administrator defines the service adapter modules, defines the data feeder object, and instantiates the data feeder instances that will collect the data.

5.1.4.1 Defining the Service Adapter Modules

The service adapters shown in Figure 28 collect data about the hardware and software of the video service. Table 5 lists the service components of each service adapter.

Table 5 Service Adapters and Service Components

Service Adapter	Service Components	Data Collected
OVIS	Platform	Current bandwidth.
	WebServer	HTTP response time.
	Dns	Average DNS look-up time, in milliseconds.
OpenView TeMIP	Platform	Platform operational state and platform pending problems.
SQL	Video	Number of downloaded movies.
	VideoServer	Timestamp from the start and end of the movie.

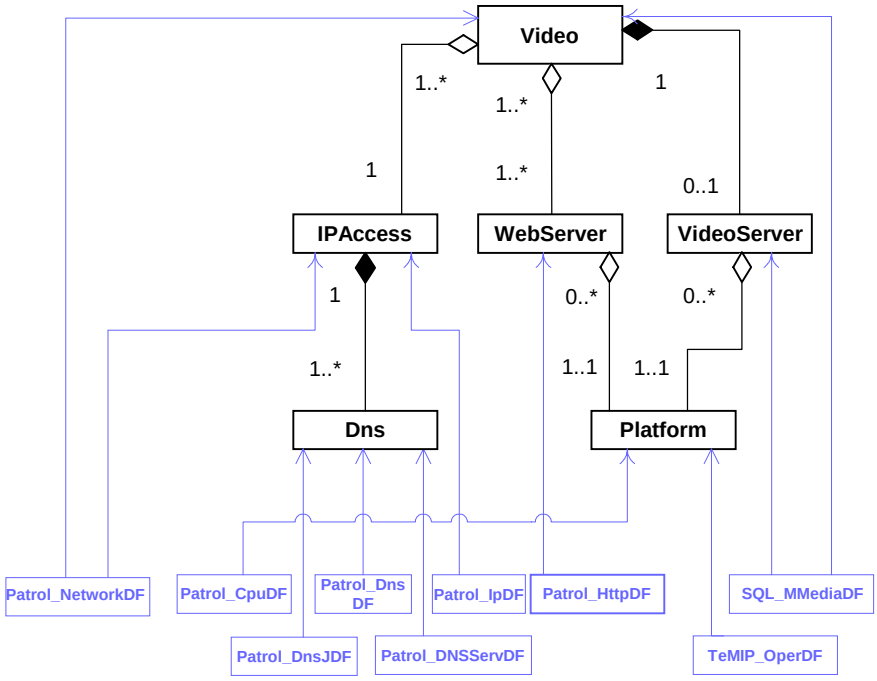
The following sections describe defining and instantiating the data feeder objects of each service adapter.

5.1.4.2 Defining the Data Feeder Objects

The ISP selects data feeders that collect data from the service component instances. Each data feeder is associated with a service adapter.

The data feeders collect information about the video service objects as shown in Figure 36.

Figure 36 Data Feeder Types



The prefix of each data feeder definition refers to its service adapter.

Table 6 describes the data feeders collecting information from the video service objects.

Table 6 Data Feeder Definition Descriptions

Data Feeder Definition Name	Description
OVIS_NetworkDF	Collects network information from the IPAccess component and the video service component.
OVIS_CpuDF	Collects CPU information from the Platform component.
OVIS_DnsJDF	Collects DNS Jarta information from the DNS component, such as the look-up delay.
OVIS_DnsDF	Collects DNS information from the DNS component.
OVIS_DnsServDF	Collects information about the state of the DNS server from the DNS component.
OVIS_IpDF	Collects IP information from the IPAccess component.
OVIS_HttpDF	Collects response time information from the WebServer component.

Data Feeder Definition Name	Description
TeMIP_OperDF	Collects operational data from the Platform component.
Oracle_MMediaDF	Collects SQL multi-media information from the VideoServer component and the video service component itself.

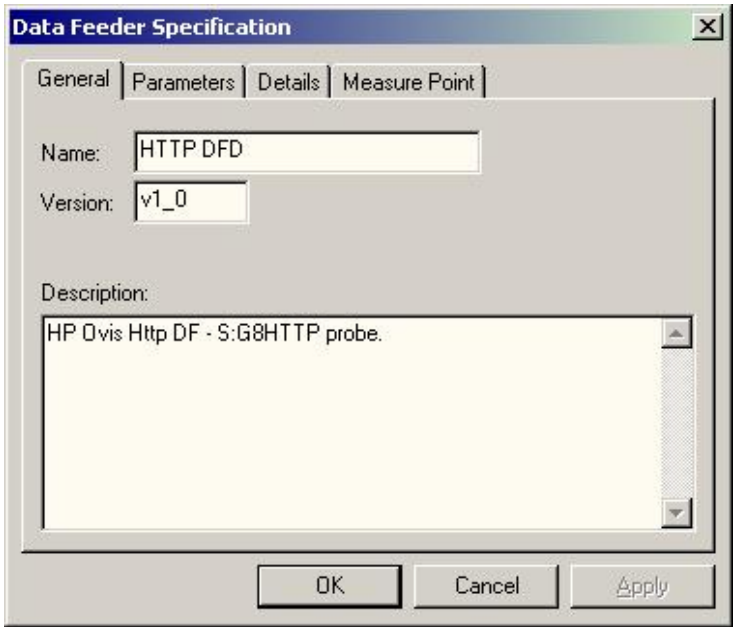
The OpenView Service Designer UI defines each type of data feeder. The definition contains a set of parameters collected by the data feeder. The ISP uses the OpenView Service Designer UI to define the following information about each type of data feeder object:

- Name of the data feeder definition, such as "OVIS_HttpDF."
- Data feeder label, such as "Http DFD."
- Name of the data feeder's service adapter, such as "OVIS."
- Properties of the monitored component, such as its name, "WebServer."
- The parameters monitored by the data feeder, such as httpResponseTime and httpRequests.
- The measure reference point (MRP) naming schema. This schema names a data feeder definition dynamically by concatenating the values of data feeder properties and fixed strings.

  For example, OpenView SQM automatically produces the name of the OVIS_HttpDF data feeder definition by concatenating the SA name with the fixed string and property provided in the data feeder definition.

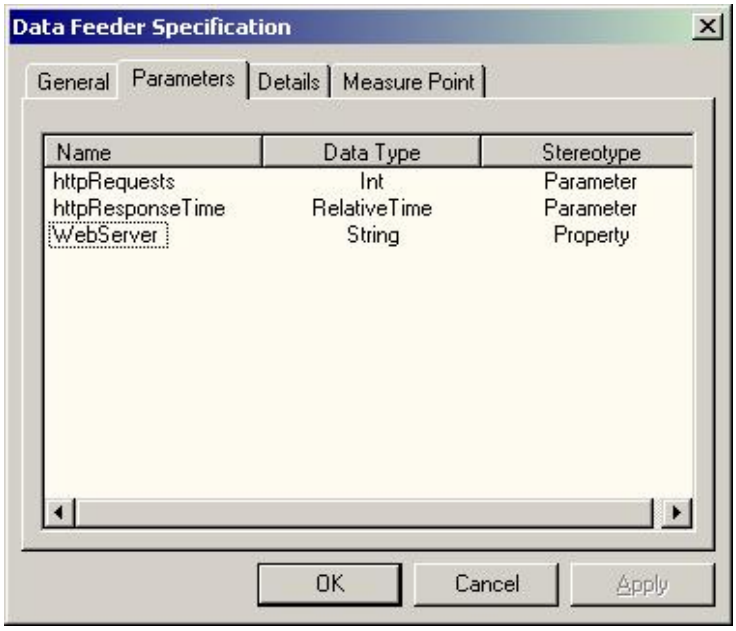
Figure 37 illustrates how the administrator creates an HTTP data feeder using the OpenView Service Designer UI.

Figure 37 OpenView Service Designer UI: Defining a Data Feeder



This data feeder is associated with a component using the Attributes tab illustrated in Figure 38.

Figure 38 OpenView Service Designer UI: Data Feeder Attributes



5.1.4.3 Instantiating the Data Feeder Objects

The administrator creates an instance of each data feeder object by associating a data feeder definition with an MRP. Table 7 describes the instances of the data feeder object the administrator creates to support the new video service.

Table 7 Data Feeder Instances

Data Feeder Definition Name	Data Feeder Instances
OVIS_NetworkDF	Net_pop_svc_Paris Net_pop_svc_NewYork
OVIS_CpuDF	Cpu_Platform-Paris Cpu_Platform-NewYork
OVIS_DnsJDF	DnsJ_16_100_200_1 DnsJ_16_100_200_2
OVIS_DnsDF	Dns_16_100_200_1 Dns_16_100_200_2
OVIS_DnsServDF	DnsServ_16_100_200_1 DnsServ_16_100_200_2
OVIS_IpDF	Ip_pop
OVIS_HttpDF	Http_webChatelet Http_webManhattan
TeMIP_OperDF	TeMIPOP_Platform-Paris TeMIPOP_Platform-NewYork
Oracle_MMediaDF	MMedia_Paris_1 MMedia_NY_1 MMedia_Paris_2 MMedia_NY_2

5.1.5 Defining Service Level Agreements

SLAs apply appropriate objectives for a given customer. OpenView SQM checks the service parameters collected by the data feeders and the service parameters calculated from the data collected against the objectives for the customer. If degraded and violated, OpenView SQM issues service alarms that generate actions, such as trouble tickets or billing rebates.

This section describes the creation of SLAs by creating service instance groups, defining the service levels, and defining the SLAs that associate service instance groups with objectives.

5.1.5.1 Creating Service Instance Groups

The SLA specifies the expected service quality using service levels. The customer will use the service through a set of service and service component instances, for example a customer in Paris will access the instances located in Paris. The administrator gathers these individual instances into a group that they can later link with the SLA.

The ISP wants to create two service instance groups, one for the service instances used by France and one for the service instances used by the United States.

To create the service instance groups, the administrator opens the OpenView SQM Administration UI and uses the Service Group dialog box. The administrator gives each service a name, a label, and associates it with one or more service instances.

The parameters described in Table 8 define the International service instance group.

Table 8 France Service Instance Group Parameters

Parameter	Value
name	"Video-International"
label	"International service instance group"
associated instances	SI name="Paris" SI name="NewYork"

The parameters described in Table 9 define the France service instance group.

Table 9 Paris Service Instance Group Parameters

Parameter	Value
Name	"France"
Label	"France service instance group"
Associated service instances	SI name="Paris"

These two service instance groups will later be associated with SLAs.

5.1.5.2 Creating Customers

An SLA is a contract between the ISP and a customer. The ISP creates two new customers using the OpenView SQM Administration UI.

The administrator defines each customer using a name, ID, and description. Two customers are created, Software House A and HP.

5.1.5.3 Defining the Service Levels

A service level defines the thresholds for controlling the quality of a service. A service level objective consists of a set of objectives for the service itself and a set of component service levels for the parameters of the service components.

The service levels define a set of parameter thresholds, a degradation factor, and an action. The component service level defines of a set of service levels that apply to a particular service component.

The video service provides three service levels, each with a different price:

- Gold-Video

  This service level provides customers with the highest possible quality of service.

- Silver-Video

  This service level provides customers with a medium-range quality of service.

- Bronze-Video

  This service level provides customers with a low-range quality of service.

The remainder of this section describes the service level objectives and component service levels created for the Gold-Video service level. The administrator would create the other service levels the same way.

Gold-Video Service Level

The Gold-Video service level defines the thresholds for the highest quality of the service. The ISP decides that the CPU load for customers of the Gold-Video service level cannot rise above 60 percent and sets the minimum bandwidth to 512 Kbps. These objectives are set using component service levels.

The administrator configures the Gold-Video service level in the OpenView Service Designer UI with the parameters given in Table 10.

Table 10 Gold-Video Service Level Parameters

Parameter	Value
Name	"Gold-Video"
Label	"Gold-Video Service Level"

The service level objectives list defines a set of parameters and gives each a threshold and an action. The Gold-Video service level has one service level objective that monitors whether the operational state is enabled. Table 11 describes the parameters of the service level objective for the operational state.

Table 11 Gold-Video Service Level Service Level Objectives

Parameter	Value	
Name	OpStateOk	
Label	Operational state is not disabled	
Description	"The operational state is enabled or idle"	
Parameter Name	OperationalState	
Action Executor	OSIGateway	
Crossing Type	Not Equal	
Threshold Levels	Action info	Critical
	Degradation Factor	100%
	Event Type	Violation
	Label	Level Violation
	Name	Level 1
	Value	Disable
	Data Type	Int

Parameter	Value	
	Clear Level Flag	True

After defining the service level objective, the administrator defines a component service level that checks the proper functioning of the platform. Table 12 describes the parameters that define the component service level.

Table 12 Gold-Video Service Level Component Objective Definition

Parameter	Value
Component service level name	"GoodPlatformHealth"
Label	"Good platform health "
SC name	"Platform"
Description	"The platform must maintain a good level of performance"

This component service level contains a list of two objectives, one that sets a threshold for the CPU load and one that sets the threshold for the minimum bandwidth. The administrator defines the component service level for the CPU load using the parameters given in Table 13.

Table 13 Gold-Video CPU Load Component Objective

Parameter	Value	
Name	CPULoadBelow60	
Label	CPU load below 60 percent	
Description	"CPU is still acceptable"	
Parameter Name	CPULoad	
Action Executor	OSIGateway	
Crossing Type	Up	
Threshold Levels	Action name	Major
	Action info	Second degradation detection
	Degradation factor	70%
	Event type	Degradation
	Label	Level Degradation
	Name	Level 1
	Value	50
	Data type	int

The administrator defines the component service level that sets the threshold for minimum bandwidth using the parameters given in Table 14.

Table 14 Gold-Service Minimum Bandwidth Component Objective

Parameter	Value
Name	MinimumBandwidth512

Parameter	Value	
Label	Minimum Bandwidth 512 Kbps	
Description	"Bandwidth must be higher than 512 kbps"	
Parameter Name	Bandwidth	
Action Executor	OSIGateway	
Crossing Type	Down	
Threshold Levels	Action name	Critical
	Action info	Violation detection
	Degradation factor	50%
	Event type	Violation
	Label	Level Violation
	Name	Level 1
	Value	512
	Data type	float

5.1.5.4 Defining the Service Level Agreements

Once the administrator has defined the service levels, they can be used to define service level agreements. A service provider can also create operational SLAs that monitor overall network performance or help desk performance.

The ISP decides to create the following service level agreements:

- GoldVideoFR.

  This is a customer SLA designed for Software House A, a customer created in section 5.1.5.2. Because Software House A has international offices, it is part of the Video-International service instance group defined in Section 5.1.5.1. The service level for this SLA is the Gold-Video service level.

- SilverVideoIDF.

  This is an operational SLA used to verify network degradation in the Paris metropolitan area, which receives the bulk of customer traffic. Because this SLA needs to monitor only Paris data, it is associated with the Video-France service instance group defined in Section 5.1.5.1. The service level associated with this SLA is the Silver-Video service level.

- BronzeVideoFR.

  This is a customer SLA designed for HP, another customer of the ISP. Because HP also has offices internationally, it is part of the Video-International service instance group defined in Section 5.1.5.1. The service level associated with this SLA is the Bronze-Video service level.

5.1.6 Monitoring the Quality of Service

Now that the administrator has designed and implemented the video service, OpenView SQM manages it. The following sections describe possible contract implementation and quality of service management scenarios for the video service.

5.1.6.1 Customer SLA Violation with Alarm Generation

This scenario describes how OpenView SQM sends an alarm following a service level violation.

The customer care agent at the ISP uses OpenView SQM to measure SLA compliance of Software House A in real-time. The administrator defines a GoldVideoFR SLA in OpenView SQM. The resources that provide measures and service component status are running and available.

The customer care agent monitors the parameter values defined in the SLA using the OpenView SQM Monitoring UI. OpenView SQM validates the parameter values against the objective parameters and publishes the results as messages.

An SLA violation begins when the CPU threshold rises over the 60 percent identified in the service level. OpenView SQM generates an alarm and publishes messages to any system involved in further processing.

5.1.6.2 Quality of Service Management

This scenario describes how OpenView SQM monitors network performance.

The ISP uses OpenView SQM to monitor in real-time the quality of the services from both its own perspective (using the operational SLA) and the customer perspective (using the customer SLAs).

An operator at the ISP monitors the average transit delay across the service network to the service provider router using the OpenView SQM Monitoring UI. The probes and performance tools are running and available. An administrator has defined the service in OpenView SQM, and defined the threshold for particular SLAs.

OpenView SQM monitors the performance values for defined SLAs. It validates the parameter values against objective parameters and publishes the results as messages. OpenView SQM notifies field engineers if a delay occurs and takes corrective actions to improve the network performance.

Appendix A

Acronyms

The following table describes the acronyms commonly used in this document:

Term	Description
ATM	Asynchronous Transfer Mode
BI	Business Intelligence
BO	Business Object
BSS	Billing Support System
CNM	Customer Network Management
CRM	Customer Relationship Management
DC	Data Collector
DFD	Data Feeder Definition
DFI	Data Feeder Instance
GUI	Graphical User Interface
IP	Internet Protocol
ISP	Internet Service Provider
IT	Information Technology
LAN	Local Area Network
MRP	Measurement Reference Point
MSL	Management Specification Language
MTBF	Mean Time Between Faults
MTTR	Mean Time To Repair
MVNO	Mobile Virtual Network Operator
NOC	Network Operation Center
OS	Operating System
OSI	Open Systems Interconnection
OSA	Open Service Architecture
OSS	Operation Support System
QoS	Quality of Service
SA	Service Adapter

Term	Description
SAI	Service Adapter Instance
SAP	Service Access Point
SC	Service Component
SD	Service Definition
SI	Service Instance
SIA	Service Impact Analysis
SIG	Service Instance Group
SIM	Service Integration Map
SLA	Service Level Agreement
SLM	Service Level Management
SLO	Service Level Objective
SNMP	Simple Network Management Protocol
SOC	Service Operation Center
SPD	Service Parameter Definition
SPDM	Service Performance Data Manager
SR	Service Resource
SRM	Service Repository Manager
SQL	Standard Query Language
TeMIP	Telecom Management Information Platform
TOM	Telecommunications Operation Map
UI	User Interface
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
XML	eXtensible Mark-up Language
WAN	Wide Area Network
WAP	Wireless Access Protocol

Appendix B

Standards Conformance

OpenView SQM relies on and is conformant with the standards from TMF and ITU. This appendix describes the relevant documents from each organization.

TeleManagement Forum Standards

The TeleManagement Forum (TMF) is a non-profit global organization that provides leadership, strategies, and solutions to improve the management and operation of communications services. The OpenView SQM design relies on the *Service Level Agreement Management Handbook*, BB 917, v1.5, published by the TeleManagement Forum.

ITU Standards

The International Telecommunications Union (ITU) consists of governments and the private sector working together to coordinate the operation of telecommunication networks and services. The ITU advances the development of communications technology. The OpenView SQM design relies on *the Quality of Service, Network Management and Traffic Engineering* (E. 800) document.

Appendix C

Basic UML Overview

The Unified Modeling Language (UML) is a visual modeling language for specifying, making, and documenting the artifacts of a software-intensive system. It helps you model software systems and distributed applications. For this reason, OpenView SQM uses UML to model services.

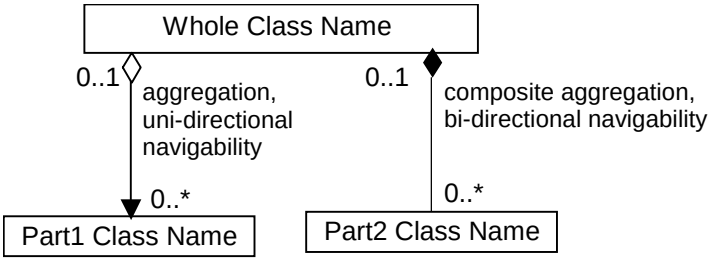
This appendix describes the UML conventions used by OpenView SQM. It contains the following sections:

- About class diagrams.
- About interaction diagrams.

About Class Diagrams

Class diagrams describe the types of object in a system and their relationships in a logical view. Figure 39 illustrates an aggregation, navigability, and multiplicity class diagram.

Figure 39 UML Class Diagram



The following sections describe the symbols used in the class diagram in more detail.

About Associations

Associations represent relationships between instances. They can be bi-directional, meaning they can be navigated in either direction, or uni-directional, meaning they can be navigated in only one direction. An arrow represents the direction of navigation. A connecting line with no arrow, as illustrated in Figure 39, represents bi-directional navigability.

About Cardinality

Associations have cardinality, meaning they specify how many instances of each component object can be present. Examples of cardinality notation follow:

0..1	Zero or one
1..*	One to many
0..*	Zero to many

An asterisk (*) represents an unlimited boundary.

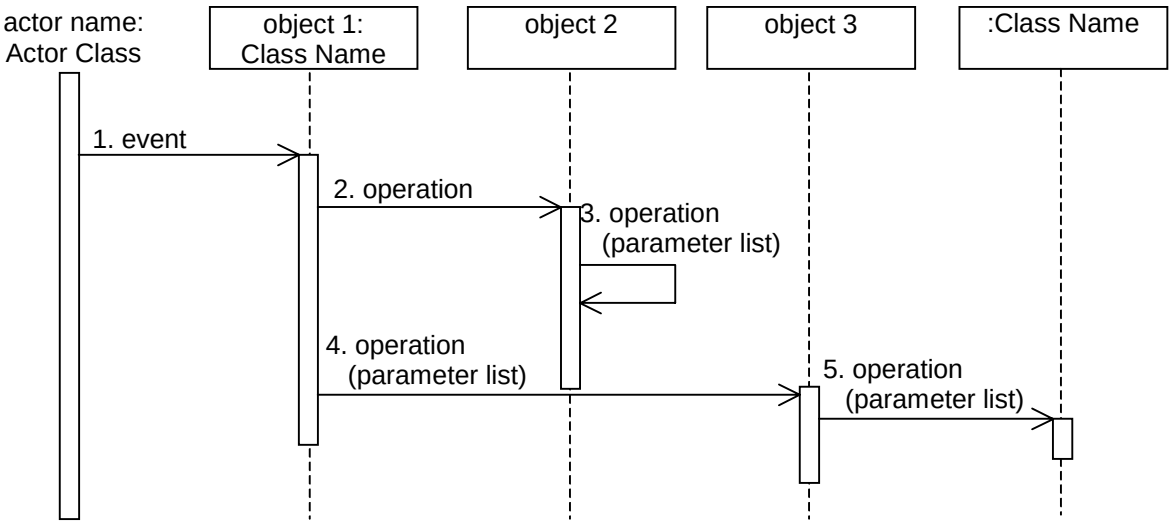
About Aggregations

As shown in Figure 39, empty and filled diamonds represent aggregations. In UML, an empty diamond aggregation means that the whole maintains a reference to its parts, so that the whole may not have created the part. The filled diamond means that the whole is responsible for creating its parts.

About Interaction Diagrams

Interaction diagrams show objects in the systems and how they interact. Figure 40 illustrates a UML sequence interaction diagram.

Figure 40 UML Interaction Diagram



In the diagram, the vertical axis represents time and the horizontal axis represents objects. Objects can self-delegate (as show in operation three), iterate, create new items (as shown as the result of operation five), and return values.

Glossary

This glossary defines terminology commonly used in HP 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 the 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 the total failure of the *service*.

Instantiate (SLA Administration)

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

measurement interval

The interval of time 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

U

V

X