HP OpenView
Service Quality Manager

SQM, TeMIP, and Acanthis KnowledgeWare

Integration Cookbook

Edition: 1.2

December 2004

Legal notices

Origin

Printed in France.

Contents

Preface

This document describes the steps that you must perform to feed an HP OpenView Service Quality Manager (SQM) application with key primary data collected by an existing HP OpenView Telecommunication Management Information Platform (TeMIP).

These mainly consist in a series of configuration tips and hints for use on the TeMIP, Acanthis KnowledgeWare and SQM production platforms (each of which is installed and configured independently). These platforms run under the HP implementation of UNIX®, HP-UX. HP-UX, which is compatible with various industry standards, is based on the UNIX System V Release 4 operating system and includes important features from the Fourth Berkeley Software Distribution.

In performing these instructions you therefore customize SQM and TeMIP so that they interoperate. This enables SQM to generate Key Performance Indicator (KPI) measures that integrate TeMIP collected key data. The integrated solution uses custom SQM SQL Service Adapters to integrate the data in an SQL database.

This document explains how to:

- Configure a TeMIP platform to collect key measures and valuable data from various sources.
- Extend an existing TeMIP configuration by integrating the Acanthis KnowledgeWare solution set to export raw or added-value TeMIP data to an external SQL database.
- Customize an SQL Service Adapter to interface SQM with this external SQL database.
- Perform administrative tasks relating to TeMIP–SQM interoperation.

Intended audience

This document is intended for Service Quality Manager Administrators and Integrators.

Required knowledge

Service Quality Manager Administrators and Integrators are expected to be familiar with the functionality of Service Quality Manager, TeMIP, and Acanthis KnowledgeWare, and have previous experience of:

- System administration and operations
- SQL database administration and operation
- Service Level Management

Service Quality Manager Administrators and Integrators are also expected to be familiar with the concepts described in the following books:

- *HP OpenView Service Quality Manager Overview*
- *HP OpenView Service Quality Manager Administration Guide*
- *HP OpenView Service Quality Manager SQL Service Adapter Toolkit Installation, Configuration and User's Guide*
- *HP OpenView TeMIP Overview & Concepts Guide*
- *Acanthis KnowledgeWare User's Guide*
- *Acanthis Attribute Exporter FM User's Guide*
- *Acanthis Notification Exporter FM User's Guide*
- *Acanthis Extended Archiving FM User's Guide*

Software versions

The software versions referred to in this document are specified in the *HP OpenView Service Quality Manager Overview* and the *HP OpenView TeMIP Overview*.

Typographical conventions

`Courier` Font:

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

Italic Text:

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

Bold Text:

- Introduces new terms and emphasizes important words

Associated documents

For a full list of the Service Quality Manager user documentation, see the *HP OpenView Service Quality Manager Overview*.

For a full list of TeMIP user documentation, see the *HP OpenView TeMIP Overview*.

Support

Please visit our HP OpenView web site at: openview.hp.com/

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

The HP OpenView support area of the HP OpenView web site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting

- Training information
- Support program information

Chapter 1

Introduction

This document assumes that you already have a detailed specification defining which relevant data (key performance indicators) your solution should process.

This document is intended only to describe HP OpenView TeMIP, Acanthis KnowledgeWare, and the HP OpenView SQM collection models that should be used. The processes and hints contained in this cookbook do not provide information on data modeling matters.

This integrated solution processes data in three stages as follows:

1. The TeMIP platform collects data from key network and service elements. Every TeMIP platform is made up of Access and Functional Modules, each one of which is responsible for handling a set of managed entities. The collected data contains information on the attributes of any entity, entity related OSI events, and alarm objects.
2. Some of these subsets of collected data are then selected as input data for the SQM platform. The Acanthis KnowledgeWare product is therefore supplied with a set of specialized Management Modules for exporting this data into an Oracle database. These modules can be used to perform five levels of export: real time, polling, prepackaged transformation by class, generic, or dedicated.
3. The SQL Service Adapters on the SQM platform collect dedicated database views from the TeMIP–SQM integrated solution. The system therefore feeds the data TeMIP collects into the SQM system through the database.

This chapter briefly describes the TeMIP and Acanthis KnowledgeWare product aspects of this integrated solution. In this integrated platform, the TeMIP and Acanthis KnowledgeWare products together form the front-end interface responsible for data collection (also called ‘acquisition’).

The SQM SQL Service Adapters then retrieve this data from the database and feed it into the SQM system as described in the final section of this chapter.

1.1 TeMIP platform

TeMIP is a distributed, scalable and model-based system in which Management Modules are used to provide the operator with network and fault management services.

1.1.1 Data collection

Some TeMIP Management Modules, called Access Modules (AMs), provide access to network elements, whereas others, called Functional Modules (FMs), perform added-value services such as monitoring the faults, state, and performance of each entity the system manages.

Access Modules are supplied with a managed entity model that includes attributes that are organized into partitions, and events that are triggered when something changes in a network element.

Functional Modules are mainly used to provide fault related services on both collected and modeled data. These services include performing alarm management, state management, network discovery, and data correlation tasks. An integrated solution typically includes the Alarm Handling FM, Notification FM, Expert System FM, and State Mapper FM, as well as other modules.

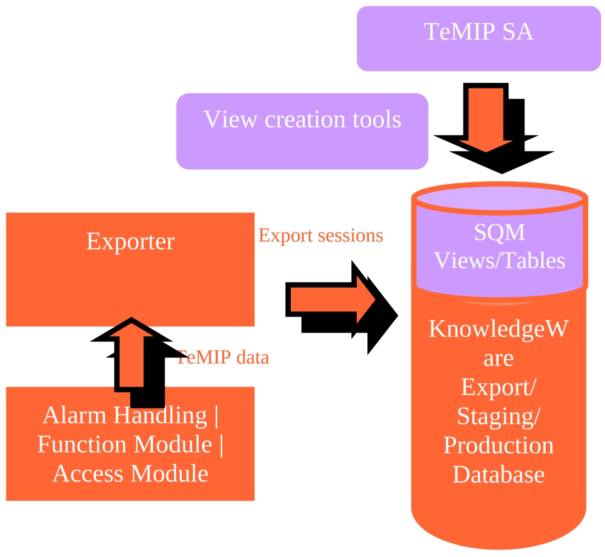
TeMIP therefore collects key data consisting in attributes, events and alarms relating to any entity modeled within TeMIP.

1.1.2 Data export

The Acanthis KnowledgeWare solution set can extract and export any TeMIP data periodically into an Oracle database. There are many potential situations in which you can usefully incorporate this software in the integrated solution, depending on which data is exported (alarms, events, attributes, performance indicators, etc.)

TeMIP data, which is in a standard format, can be integrated in SQM through the Acanthis KnowledgeWare SQL databases in various ways. You must use more than one TeMIP Service Adapter, or rather DFD, to represent and map all of the relevant TeMIP data for this reason.

You can install and configure a different SQM SQL Service Adapter (the appropriate TeMIP Service Adaptor) for each TeMIP use situation more simply by using the SQM SQL SA Toolkit.


TeMIP SA
View creation tools
Exporter
Export sessions
SQM Views/Tables
KnowledgeWare Export/ Staging/ Production Database
TeMIP data
Alarm Handling | Function Module | Access Module

This Integration Cookbook highlights which TeMIP data is typically retrieved and exported. It therefore concentrates on the DFD definition and its associated MRP definition. It additionally provides details of the system configuration required in combination with the Oracle database to execute the SQL query run by the SQL Service Adapter responsible for the DFD concerned. It also specifies the configuration required by the TeMIP Acanthis Export Modules used.

Identifying what development, integration and configuration tasks are needed for the SQM TeMIP Service Adapter to meet the operational requirements is therefore key in a successful implementation.

1.1.2.1 Database requirements

Supported operating systems

TeMIP v5.x runs on the following operating systems:

- HP-UX 11.11
- Sun Solaris 9
- HP Tru64 5.1

The integrated system documented in this Integration Cookbook was implemented with SQM running under HP-UX and TeMIP running under Tru64, for system availability reasons. You can use other systems, however, depending on which Third Party Product versions are supported.

Supported SQL databases

SQL Service Adapter V1.2 supports the following SQL databases:

- ORACLE 8i and 9i

The integrated system documented in this Integration Cookbook was created with an Oracle 9i server, which is the only database supported by the Acanthis KnowledgeWare solution set.

Supported SQL data types

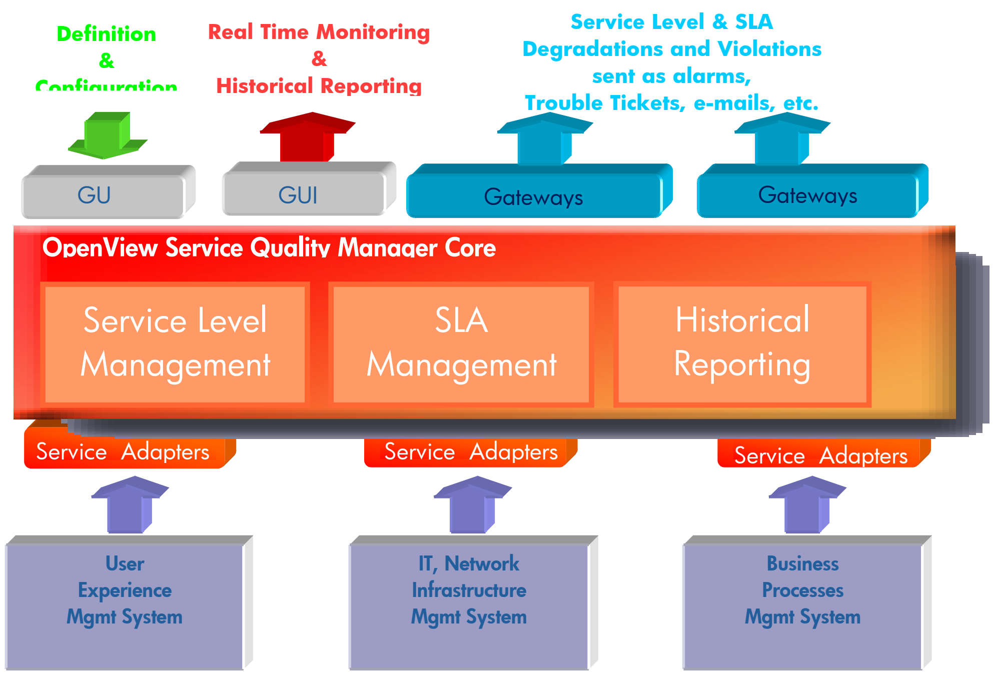
The table in appendix B lists all column data types supported by the Acanthis KnowledgeWare Exporting Functional Modules, together with their associated mapping in the Oracle database.

1.2 Service Quality Manager platform

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

SQM monitors the service quality by aggregating information coming from all data sources, such as the network, IT infrastructure, and the service provider's business processes. Using this information, service operators can pinpoint infrastructure problems and identify their potential effects on customers, services, and service level agreements (SLAs).

Figure 1 Service Quality Manager Main Components


Definition & Configuration
Real Time Monitoring & Historical Reporting
Service Level & SLA Degradations and Violations sent as alarms, Trouble Tickets, e-mails, etc.
GU
GUI
Gateways
Gateways
OpenView Service Quality Manager Core
Service Level Management
SLA Management
Historical Reporting
Service Adapters
Service Adapters
Service Adapters
User Experience Mgmt System
IT, Network Infrastructure Mgmt System
Business Processes Mgmt System

The Service Adapters, and the SQL Service Adapter in particular, provide a southbound entry interface to the SQM core. You must configure these adapters (at SQM platform level) so that they connect to a SQL database. You must also define the mapping between the collected data stored within an SQL database and the SQM data model.

For a detailed description of SQM, see the *HP OpenView Service Quality Manager Overview*.

1.2.1 SQL Service Adapter

An SQL Service Adapter is a generic Service Adapter that collects data from a given SQL database. See the SQL SA customization process section of the *HP OpenView SQL Service Adapter Toolkit Installation, Configuration and User's Guide* for more information on how to configure this acquisition layer.

The SQL Service Adapter serves as a bridge between SQM and an SQL database by doing the following:

- Collecting QoS data from SQL tables
- Exposing this data to the SQM core as SQM performance values

You must specify which method and parameters are used to process the values from an SQL table column into the SQM data model by configuring the SQL Service Adapter.

1.2.1.1 SQM data model

The following paragraphs summarize the main definitions of Service Adapter terms as applied to the SQL Service Adapter. For more information about these important SQM concepts, see the *HP OpenView Service Adapters User's Guide*.

Data Feeder Definition

A Data Feeder Definition (DFD) serves to identify a specific measurement point that provides a set of indicators (called 'parameters') on the quality aspects of a given resource type (the Response Time at a given service access point, for example). Data Feeder Definitions therefore contain a logical description of all parameters an SQL table feeds to the SQM.

An SQL Service Adapter can provide one or more Data Feeder Definitions, each of which is linked to an SQL database table.

Each Data Feeder Definition must contain the following information:

- A name
- A version
- A set of DFD parameters

  A parameter may be global (meaning it is not linked to any subscriber information) or customer specific (meaning it is collected for a specific customer). Customer specific parameters can hold different values for different customers.

  In the case of the SQL Service Adapter, a parameter correlates to the SQL table column from which the parameter values are extracted.

  If a parameter is customer specific, another column in the same SQL table must provide the customer or subscriber information associated with the collected data.
- A set of properties

  Properties are not parameters (that is, they are not performance indicators). They identify the context of the collected parameters instead. The property value is set when the data feeder instance is configured.
- A Measurement Reference Point (MRP) naming schema

  The MRP naming schema is used to build a unique identifier (by concatenating DFD properties) for each measurement point.

Data Feeder Instance

A Data Feeder Instance (DFI) publishes a specific measurement point's collected values according to a Data Feeder Definition.

When a DFD parameter is customer specific, the DFI collects the perceived QoS information from a MRP for any of the specific customer's subscribers.

For information on how to create Data Feeder Instances, see the *HP OpenView SQL Service Adapter Toolkit Installation, Configuration and User's Guide.*

Timestamp management

When the SQL Service Adapter collects information as standard, the SQL database must contain timestamp information, and you must specify its location in the database when you customize the system. If it does not provide a timestamp, you can use an SQL view or script to create a specific column containing this information. For more information, see the advanced customization chapter of the *HP OpenView SQL Service Adapter Toolkit Installation, Configuration and User's Guide*.

The SQL Service Adapter uses the timestamp directly to publish collected values and identify the latest values entered into the database (according to the difference between the timestamp and the last timestamp collected).

Customer, and subscriber

As already said, the platform can collect DFD parameter values for a specific customer (an organization or corporation) or for a specific member of that customer (a user or a subscriber). The SQL Service Adapter can retrieve QoS data for a customer or a subscriber directly.

The customer or subscriber information must be stored in a column in the DFD's SQL table.

OpenView SQM uses subscriber IDs to link subscribers with customers. These IDs identify the user of the service in a service quality measurement. The subscriber ID can be an IP address, a user login name, or an identifier directly associated with a user, such as a DSL port number for example.

In some cases, the user is also the customer. The subscriber ID then maps directly to the customer ID. This requires no further mapping, because the default domain is the generic "ServiceCenter" domain.

A mapping structure links a subscriber ID to a customer. This structure is stored in the SQM Central Repository. The structure contains the following information:

- Customer name
- Naming plan

A naming plan specifies all customer IDs in each subdomain. A domain logically groups all customer IDs.

An example of mapping is shown below.

```
Customer Name: MyNewCustomer
Naming Planes:
{
Domain Identifier: IP
ipaddress: {"16.18.*.*", "16.23.*.*"}
}
{
Domain Identifier: IMSI
ipaddress: {"12202???", "2002???"}
}
{
Domain Identifier: Mail
ipaddress: {"*@hp.com"}
}
```

For more information on subscriber ID mapping, see the *SQM Information Modeling Reference Guide*.

Note

Customer or subscriber specific parameters require one column of the DFD table to contain the customer or subscriber information.

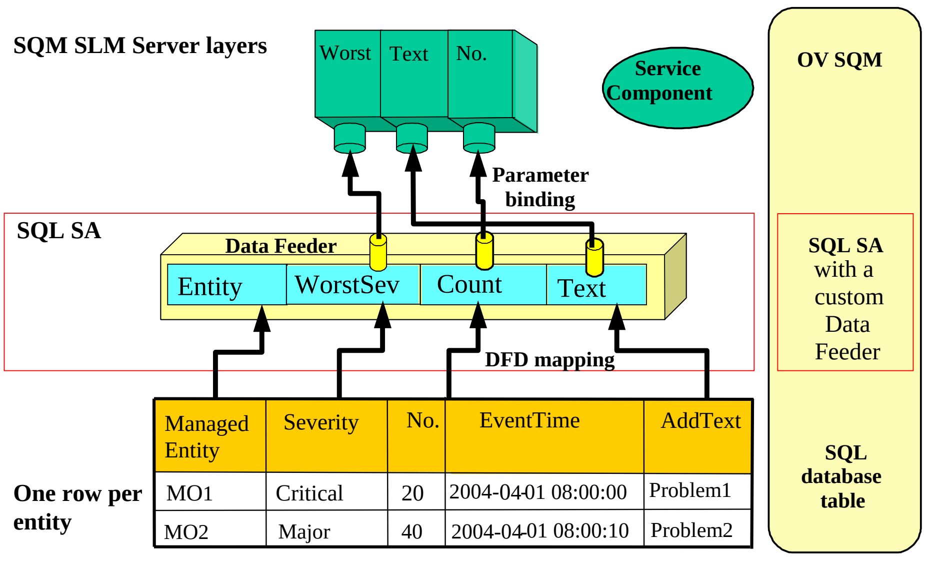
If the table contains a subscriber column, the column must contain subscriber IDs for a single subscriber domain.

1.2.1.2 Data collection overview

The DFD and SQL database table fields must be mapped straightforwardly and according to the following rules:

- Each DFD must be linked to one SQL table
- Each DFD parameter must be linked to one table column
- Each MPR property must be linked to one table column
- Each MRP can contain several DFD properties

If the database schema does not offer these mapping rules, you must consider using SQL views instead. For information on creating SQL views, see the appropriate chapter in the *HP OpenView SQL Service Adapter Toolkit Installation, Configuration and User's Guide.*



The SQL Service Adapter (SQL SA) retrieves data from a database. It uses the Java™ Database Connectivity (JDBC) drivers provided by the database's manufacturers to establish connections to the databases. The application then uses these connections to send SQL queries that retrieve data from the tables. The SQL Service Adapter is also responsible for closing the connections.

Depending on the physical configuration and environment required in the integrated system, you must either install the Service Adapter on the SQL database server or provide remote database access. Although you can configure the integrated system in either way, you must bear the following points in mind when you choose the most suitable configuration for your particular installation:

- If the database is accessed remotely, you must validate the connection to the dedicated database port.
- If the Service Adapter can only be accessed locally, you must check which operating system the SQL Service Adapter supports.

The SQL Service Adapter uses a polling mechanism to collect performance data. At each polling interval, the Service Adapter connects to the database to execute an SQL collection query that retrieves the data values. It then maps the collected data to the defined DFD parameters and publishes it, with timestamps, to the SQM core layer (the Service Level Monitoring Client). You must define this DFD mapping as part of the SQL SA customization process.

The system collects the *WorstSeverity* and *NbAlarms* parameters. Both these parameters are represented as table columns in the database.

The system uses the *Entity* property as the MRP to uniquely identify where the data is collected from.

1.2.1.3 Database requirements

Supported operating systems

SQL Service Adapter v1.2 runs on the following operating systems:

- HP-UX 11.11
- Windows 2000®/XP®

Supported SQL databases

SQL Service Adapter V1.2 supports the following SQL databases:

- Sybase 11.9.2
- SQL Server 2000
- MySQL 4.0.15
- Oracle 8i and 9i

Supported SQL data types

If the Service Adapter collects a parameter from a table column whose data type is not supported, you must create a special view or script to convert the column’s data type to a supported data type. For information on how to create SQL views, see the *Advanced customization* chapter of the *HP OpenView SQL Service Adapter Toolkit Installation, Configuration and User’s Guide*. If the column’s data type cannot be converted to a supported SQM data type, a dedicated function must be developed for the SQL Service Adapter so that it supports this special type. To request this custom development, contact your HP Sales Representative.

Chapter 2

Platform interoperation

SQM and TeMIP interoperate on top of an SQL database. An SQL Service Adapter defines which DFD parameters and properties the SQM–TeMIP interoperation collects from the columns of a table in the database, using an SQL query.

2.1 Module deployment

All integration solutions are based on a standard TeMIP director, access server, or Full Server. You must also install all the Network Element Access Modules and the Advanced Functional Module on the TeMIP director. Lastly, you must deploy the Acanthis TeMIP Management Modules used to export the TeMIP data into an Oracle database.

The TeMIP director is supplied with a set of standard Functional Modules that include the Alarm Handling Functional Module and the Notification Functional Module. You can also choose to install additional TeMIP products, such as the Expert System Functional Module to monitor and correlate alarms, or a specialized Service Monitoring Functional Module to model and supervise services, for example.

The Acanthis KnowledgeWare solution set is supplied with a complete suite of Functional Modules that include TeMIP Data Exporter Modules, Oracle Database Monitoring Modules, a Data Warehouse Workflow Module, and additional Oracle Transformation Packages for use in specific implementations.

Depending on how you implement the customer use cases described in this chapter, you might also need to install TeMIP Data Exporter Modules. The exact combination of modules required varies according to the specific implementation concerned.

SQM does not require the Oracle Database Monitoring Modules in this integrated solution. These would re-inject data changes occurring in the database into TeMIP, notably after the data has been staged and transformed.

The Data Warehouse Workflow Module is only needed if the integrated solution handles parameters that transformation packages calculate based on the data exported by TeMIP. This data is specific to a particular field, such as IP networks, and it is obviously refreshed less often than exported data since it is produced by a much longer process in which the data is exported, staged, transformed, and aggregated.

2.2 Integrated data collection

The Acanthis Exporter Functional Modules collect data for the TeMIP system. Each module handles one specific aspect of the TeMIP entity model and the default TeMIP fault management models (attributes, events, or alarms). They do not address other aspects of the TeMIP application, such as monitoring or logging.

The integrated system uses export contexts to define how data is exported. These function differently depending on which Data Exporter Module is used.

Before you can use the Attribute Exporter FM, you must define which attributes are exported, so that all the Oracle database tables that receive the exported data are ready to use.

SQM–TeMIP interoperation uses a polling mechanism to export attributes for each entity class. It also provides a post-export function hook that you can use to perform any additional processing required by the collected data, or for other purposes. This hook performs a *show* directive on the attributes of entities that are declared as members of the export context but do not have a special scheduling policy during the polling period.

Similarly, before you can use the Notification Exporter FM, you must define which events are exported, so that all the Oracle database tables that receive the exported data are ready to use.

The Notification Exporter FM can either export events for specific entity classes or export events for all classes. The export context is event-driven, meaning that when an event is caught it is exported immediately. It too provides a post-export function hook that you can use to perform additional processing after the data has been exported. It performs a *notify* directive on its associated domain, which is defined as an export context attribute.

The Extended Archiving FM is supplied with prepackaged tables because it is based on a standard TeMIP model, the operation context alarm objects.

Its export context is based on a polling mechanism that exports all TeMIP alarms, regardless of their state, for an operation context. It uses a table to store the alarms but also uses additional tables to store alarm parameters with complex datatypes.

2.3 Platform interconnection

A key feature of the integration is that the platforms interconnect through the Oracle database. This means you must ensure the SQM SQL SA products do not overload the database, and allow the Acanthis product to perform properly. You must also minimize the effect of Acanthis product use on TeMIP platform performance.

In addition, you must not modify the existing structure of the Oracle database. This is because the use of Acanthis Data Exporter Modules may be affected if you customize the export tables.

You can integrate additional packages that work with temporary tables or simple views based on the export tables, however.

2.4 Integration licenses

The only licenses needed to implement an operational integration solution, in addition to the standard TeMIP, SQM and Oracle licenses, are the SQM SQL SA Toolkit license and the Acanthis Exporter license for the Data Exporter Module you use in the integration.

Chapter 3

TeMIP use cases

All the following TeMIP use cases have been implemented as studies and are referred to as examples throughout this Implementation Cookbook. Each is described in the context of the chapter, and then highlighted and detailed. Every use case introduces typical problems encountered when implementing an integrated solution.

3.1 Entity attributes

This use case illustrates how you can map a TeMIP entity attribute directly to an SQM parameter. It is referenced in this document as UC1–Entity Attributes.

Its usefulness is restricted by the semantics and refresh rate of the attribute being exported, and depends on whether the TeMIP Management Module can compute or update it.

Its added value is therefore limited to the SQM model that uses the parameter. The data’s granularity is either at managed entity or class level.

The TeMIP Generic State partition’s attributes are taken as an example in this use case.

3.1.1 TeMIP Generic States

Every TeMIP entity can support the Generic States partition, which contains a set of standardized status attributes. It supports it internally, either through the Access Module or Functional Module or through a TeMIP State Mapper Functional Module that maps native attributes to generic state attributes.

3.1.1.1 Definition

The Generic State partition contains the following attributes:

- Primary retrieved or computed state attributes:

Name	Values	Comment
Generic Composite Operational State	NotManaged *(0)* Testing *(1)* Unknown *(2)* Idle *(3)* Active *(4)* Busy *(5)* Unstable *(6)* Partial *(7)* Indeterminate *(8)* Disrupted *(9)* NotFunctional *(10)*	Computed automatically from other generic states

Name	Values	Comment
	FutureUse1 *(11)* FutureUse2 *(12)*	
Generic Unknown Status	*True/False*	
Generic Operational State	Disabled *(0)* Enabled *(1)*	
Generic Usage State	Idle *(0)* Active *(1)* Busy *(2)* Unknown *(3)*	
Generic Alarm Status	Indeterminate *(0)* Critical *(1)* Major *(2)* Minor *(3)* Warning *(4)* Clear *(5)*	
Generic Availability Status	InTest *(0)* Failed *(1)* PowerOff *(2)* OffLine *(3)* OffDuty *(4)* Dependency *(5)* Degraded *(6)* NotInstalled *(7)* LogFull *(8)* Available *(9)*	
Generic Administrative State	Locked *(0)* Unlocked *(1)* ShuttingDown *(2)*	
Generic Procedural Status	InitializationRequired *(0)* NotInitialized *(1)* Initializing *(2)* Reporting *(3)* Terminating *(4)* Available *(5)*	
Generic Control Status	SubjectToTest *(0)* PartOfServicesLocked *(1)* ReservedForTest *(2)* Suspended *(3)* Available *(4)*	
Generic Standby Status	HotStandBy *(0)* ColdStandBy *(1)* ProvidingService *(2)*	

- Operator defined outage state attributes:

Name	Values	Comment
Generic Managed Status	*True/False*	Set by the operator

Name	Values	Comment
Generic Testing Status	*True/False*	Set by the operator

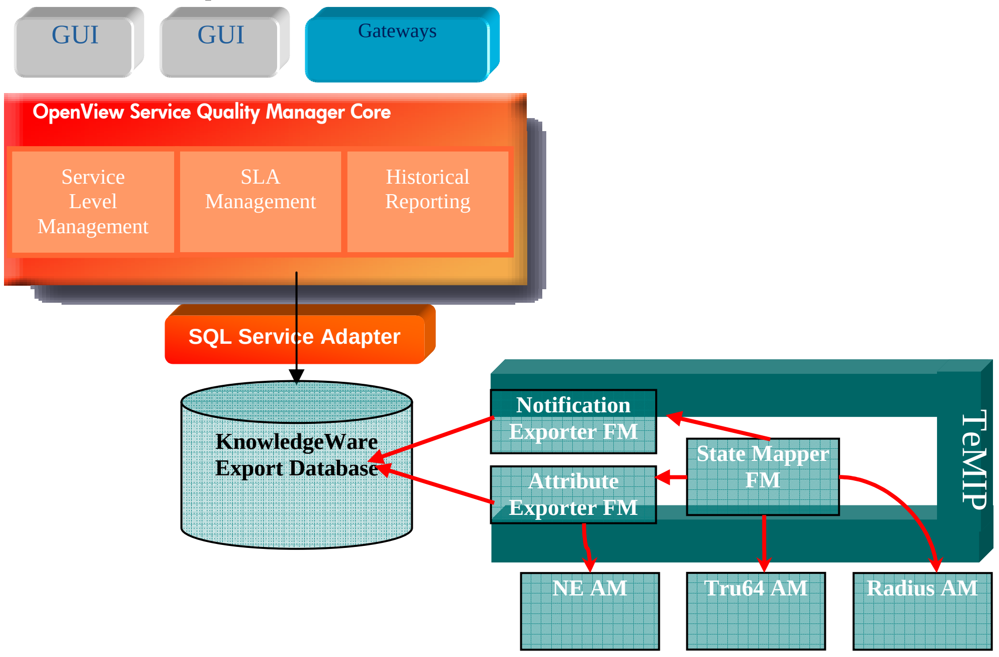
- Associated timestamps to check and provide information on state validity:

Name	Values	Comment
Generic Managed Status Change Timestamp	*AbsTime*	
Generic Testing Status Change Timestamp	*AbsTime*	
Generic Global Change Timestamp	*AbsTime*	
Generic Refresh Timestamp	*AbsTime*	

3.1.1.2 Architecture

The Acanthis Attribute Exporter FM is required to export TeMIP attributes.

The following diagram illustrates how TeMIP and SQM are integrated through the Attribute Exporter FM.



The Generic State attributes are either retrieved directly from the Access Module or Functional Module, or computed or mapped by a State Mapper Functional Module. They can then be accessed as attributes or state change events.

3.1.2 DFD definition

The TeMIP State DFD of each entity managed by the TeMIP platform has the following SQM related characteristics.

DFD Characteristics	
DFD Name	*TeMIPStateDFD*
DFD Label	*TeMIP Generic States DFD*
DFD Version	*V1_0*
SA Name	*TeMIPStateSA*
SA Version	*V1_0*

Parameter Name	**Description**	**Data Type**	**Default Value**
CompositeState	Network Element Composite Operational State	Enum	
UnknownState	Network Element Unknown State	Enum	
OperationalState	Network Element Operational State	Enum	
UsageState	Network Element Usage State	Enum	
AlarmState	Network Element Alarm Status	Enum	
AvailabilityState	Network Element Availability Status	Enum	
AdministrativeState	Network Element Administrative State	Enum	
ProceduralState	Network Element Procedural Status	Enum	
ControlState	Network Element Control Status	Enum	
StandbyState	Network Element Standby Status	Enum	
ManagedStatus	Network Element Managed Status	Enum	
TestingStatus	Network Element Testing Status	Enum	
ManagedLastTimestamp	Managed Status Change Timestamp	AbsTime	
TestingLastTimestamp	Testing Status Change Timestamp	AbsTime	
GlobalLastTimestamp	Global Change Timestamp	AbsTime	

Property Name	Description	MRP	Data Type	Default Value
EntityName	Network Element Full Entity Name	Yes	String	

3.2 Entity events

This use case illustrates how you can map specific TeMIP events to an SQM parameter. It is referenced in the document as UC2–Entity Events.

You can use it to select a set of events for mapping to a number system. It is equally useful because the events are exported immediately they are received. TeMIP is supplied with an extensive set of predefined OSI events for many classes.

Its added value lies in providing a global snapshot view of an entity, based on the set of events it has received. The data's granularity is either at managed entity, class or domain level. You can use the domain entity to customize the grouping of managed entities.

The study takes the TeMIP *IP Reachability Up* and TeMIP *IP Reachability Down* events as an example.

3.2.1 TeMIP IP 'reachability'

You can use the TeMIP IP Poller Functional Module to check the availability ("reachability") of every network element in the infrastructure. It polls the network elements to check their availability and triggers associated events to reflect that availability.

3.2.1.1 Definition

The IP availability events are as follows:

Name	Comment
IP Reachability Up	Received as soon as the Network Element can be reached again
IP Reachability Down	Received as soon as the Network Element can no longer be reached

These events contain the following attributes:

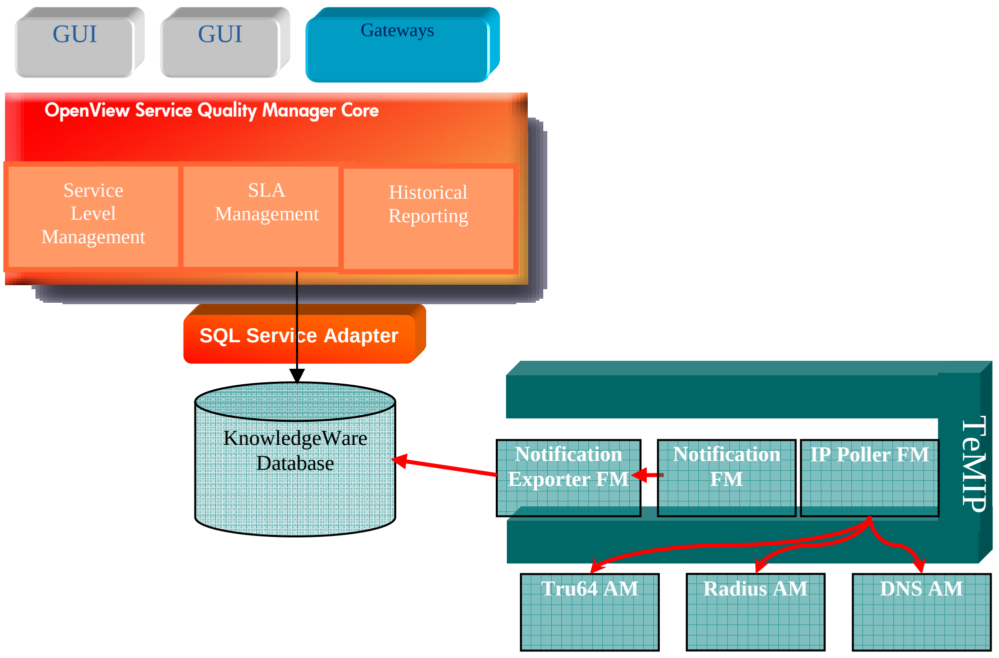
Name	Comment
Event Type	
Event Time	
Probable Cause	
Perceived Severity	
Additional Text	
Managed Object	

Only the *Event Name*, *Event Time* and the associated *Managed Object* attributes are useful in this use case.

3.2.1.2 Architecture

The Acanthis Notification Exporter FM is needed to export TeMIP events.

The following diagram illustrates how the Notification Exporter FM serves to integrate TeMIP and SQM.



The integrated system triggers IP *Reachability* events when the IP Poller FM polls the ICMP and detects that the Network Element does not respond when it is polled. Each entity managed by TeMIP must support these events together with the *GetEvent* directive.

3.2.2 DFD definition

The TeMIP Availability DFD of each entity managed by the TeMIP platform has the following SQM related characteristics.

DFD Characteristics			
DFD Name	*AvailabilityDFD*		
DFD Label	*TeMIP Availability DFD*		
DFD Version	*v1_0*		
SA Name	*TeMIPAvailSA*		
SA Version	*v1_0*		
Parameter Name	**Description**	**Data Type**	**Default Value**
Availability	Network Element Availability	Enum	Available

Property Name	Description	MRP	Data Type	Default Value
EntityName	Network Element Full Entity Name	Yes	String	

3.3 Alarm statistics

This use case shows how you can calculate SQM parameters from alarm statistics. It is described in more detail in the *SQM TeMIP Fault SA Specification*. It is referenced in the document as UC3–Alarm Statistics.

You can use it to provide a set of parameters calculated from a set of alarms on a set of entities or on each entity. Most network operators deal only with alarms collected in operation contexts. It is particularly useful in providing them with statistics calculated from what they observe or do when these alarms occur, because this truly represents the network management view.

This use case offers a broad range of potential granularities. It can focus on a specific entity, all entities in the same class, a specific set of entities, or all entities in a domain, for example.

It can also be used to represent all TeMIP domains within SQM, where these are used to provide a geographical, manufacturing, technology or other grouping. Its aim is not to represent the entire domain hierarchy in this scenario but to represent key domains instead, so that the definitions can at least be used to populate or upload the data.

The study takes the TeMIP alarms as an example, because the first use case deals with using the operation context attributes to produce TeMIP Fault statistics, while the second use case deals with using operation context events to produce TeMIP Fault statistics.

3.3.1 TeMIP operation context

A TeMIP operation context serves to group together all TeMIP alarm objects created when OSI alarm events are collected from TeMIP managed entities belonging to the operation context’s associated domain. The operation context provides a set of services focused on the alarm objects, including lifecycle, archiving and similarity management.

A TeMIP operation context includes many attributes reflecting its characteristics, states and added value counters. Although such TeMIP fault statistics may be useful, they relate to one operation context only and cannot be used when the TeMIP platform collects fault statistics data for each entity managed by the TeMIP system.

3.3.2 TeMIP alarm object

An operation context serves to group together all TeMIP alarms on every managed entity in the operation context’s related domain. Several operation contexts, each of which is handled by a different operator, can receive the same OSI alarm event. Each operation context therefore contains an alarm object. The alarm object is associated with the operation context, meaning it can be handled entirely differently depending on the operator’s context and role.

The Access Modules issue the original OSI events, which are then collected by the Alarm Handling Functional Module. The Alarm Handling Repository then stores these events as alarm objects.

3.3.2.1 Definition

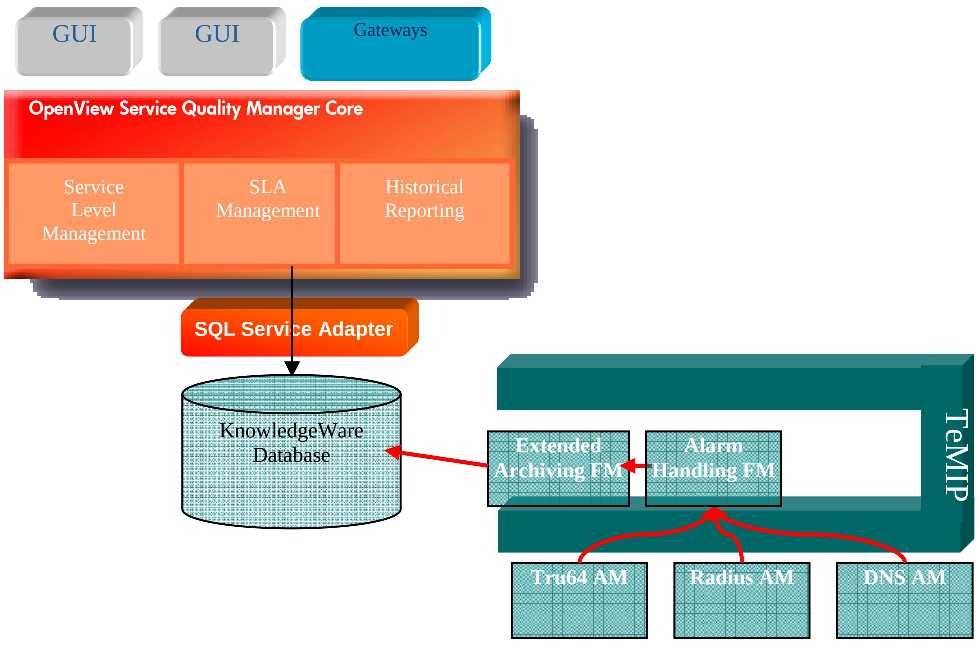
A TeMIP alarm is primarily composed of the following:

Name	Values	Comment
Identifier	Integer	Assigned to the alarm object for the operational context
Managed object	*FullEntityName*	Name of the managed entity
Alarm type	*CommunicationsAlarm* *EnvironmentalAlarm* *EquipmentAlarm* *IntegrityViolation* *OperationalViolation* *PhysicalViolation* *ProcessingErrorAlarm* *QualityofServiceAlarm* *SecurityServiceOrMechanismViolation* *TimeDomainViolation*	Type of alarm
Perceived severity	*Indeterminate* *Clear* *Warning* *Minor* *Major* *Critical*	Alarm severity
State	*Outstanding* *Acknowledged* *Terminated* *Archived*	Status of the alarm object
Event Time	AbsTime	Date and time the managed entity received the alarm event
Creation Timestamp	AbsTime	Date and time the alarm object was created in the operation context
Probable Cause	Customizable enumeration type. Many default values.	Set of normalized/ predefined causes of the alarm
Additional Text	String	Text linked to the original alarm event

3.3.2.2 Architecture

The Acanthis Extended Archiving FM is needed to export TeMIP alarms.

The following diagram illustrates how the Extended Archiving FM is used to integrate TeMIP and SQM.



The TeMIP platform creates an alarm object when it receives an OSI alarm event.

3.3.3 DFD definition

The TeMIP Fault Statistics DFD of each entity the TeMIP platform manages has the following SQM related characteristics.

DFD Characteristics			
DFD Name	*TeMIPFaultStatsDFD*		
DFD Label	*TeMIP Fault Statistics DFD*		
DFD Version	*v1_0*		
SA Name	*TeMIPFaultStatsSA*		
SA Version	*v1_0*		
Parameter Name	**Description**	**Data Type**	**Default Value**
WorstSev	Worst alarm severity	Int	*0*
WorstAddText	Worst Additional Text	String	No text available
NbOutstandingAlarm	Number of outstanding alarms	Int	*0*
NbIndeterminateAlarm	Number of indeterminate alarms	Int	*0*
NbWarningAlarm	Number of warning alarms	Int	*0*
NbMinorAlarm	Number of minor alarms	Int	*0*

NbMajorAlarm	Number of major alarms	Int	*0*
NbCriticalAlarm	Number of critical alarms	Int	*0*
ProbDuration	Problem duration	Rela-tiveTime	*00:00:00*
LastSeverity	Last known severity	Int	*0*
LastAddText	Last known additional text	String	No text available
LastTimestamp	Last alarm timestamp	AbsTime	Now

Property Name	**Description**	**MRP**	**Data Type**	**Default Value**
EntityName	Network Element Full Entity Name	*Yes*	String	
OperationContext	Operation Context collecting alarms on managed object	*Yes* | *No*	String	
CollectionDomain	Domain containg as a member the managed object and used by the operation context	*Yes* | *No*	String	

3.4 Performance indicators

This use case shows how you can extract performance indicators from alarms into SQM parameters. It is referenced in the document as UC4–Performance Indicators.

It can be used to extract key performance indicators that are not included in the TeMIP entity model from OSI events. By extension, it can also be used to retrieve data from event fields such as additional text and monitored attributes containing formatted data. These event fields are used in certain OSI configuration events, such as state, relationship, and attribute value change events.

It can also be useful when TeMIP does not manage the data concerned directly because it is provided by the Third Party Product or network element manager TeMIP is interfacing with instead. The data's granularity is at the managed entity level.

The study takes TeMIP alarms containing performance indicator values as an example. The *Additional Text* attribute of alarms can be processed in the same way if it contains necessary information, however.

3.4.1 TeMIP monitored attributes

TeMIP extracts these attributes from the *Monitored Attributes* OSI alarm events field.

3.4.1.1 Definition

The following performance indicators are extracted from the alarms in this example.

- From the Radius MIB:

Name	Comment
CPU Utilization	
CPU Threshold	
File System Free Space	
File System Threshold	
Response Time	
Response Time Threshold	
Queue Size	
Queue Size Threshold	

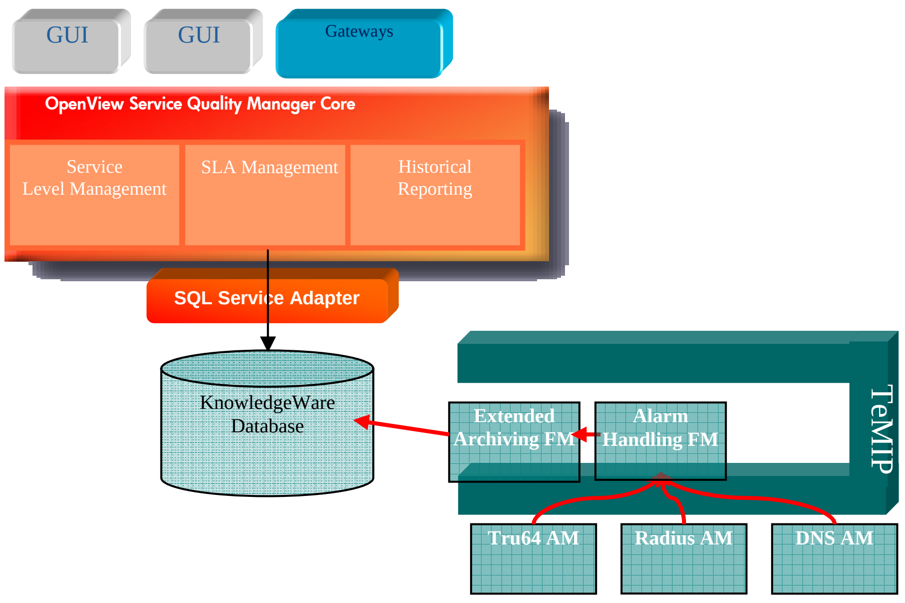
- From the DNS MIB:

Name	Comment
Response Time	
Server Address	
Severity	

3.4.1.2 Architecture

The Acanthis Extended Archiving FM is needed to export TeMIP monitored attributes.

The following diagram illustrates how the Extended Archiving FM is used to integrate TeMIP and SQM.



In this use case, you must extract and format the monitored attributes from the TeMIP alarms received for a given entity class.

3.4.2 DFD definition

The TeMIP Radius Performance DFD of each entity managed by the TeMIP platform has the following SQM related characteristics:

DFD Characteristics				
DFD Name	*RadiusDFD*			
DFD Label	*Radius Performance DFD*			
DFD Version	*V1_0*			
SA Name	*RadiusSA*			
SA Version	*V1_0*			
Parameter Name	**Description**		**Data Type**	**Default Value**
CPUUse	CPU Utilization		Float	
CPUMax	CPU Threshold		Float	
FreeSpace	File System Free Space		Float	
FreeSpaceMin	File System Free Space Threshold		Float	
RespTime	Response Time		Float	
RespTimeMax	Response Time Threshold		Float	
QueueSize	Queue Size		Int	
QueueMax	Queue Size Threshold		Int	
Property Name	**Description**	**MRP**	**Data Type**	**Default Value**
EntityName	Network Element Full Entity Name	Yes	String	

DFD Characteristics			
DFD Name	*RadiusDFD*		
DFD Label	*Radius Performance DFD*		
DFD Version	*V1_0*		
SA Name	*RadiusSA*		
SA Version	*V1_0*		
Parameter Name	**Description**	**Data Type**	**Default Value**
RespTime	Response Time	Float	
ServAddress	Server Address	String	
Severity	Severity	Int	

Property Name	Description	MRP	Data Type	Default Value
EntityName	Network Element Full Entity Name	Yes	String	

3.5 Event attributes

This use case shows how you can extract attribute values from OSI events into SQM parameters. It is referenced in the document as UC5–Event Attributes.

It can be used to provide an alternative method of polling a managed entity’s attributes. Events are asynchronous, and they only provide information on changes to an entity’s attributes. There is therefore no need for the TeMIP platform to systematically poll all attributes (including those that do not change). There is also no need for it to wait until the end of each export polling period before it retrieves updated information.

To avoid polling and checking for Generic State attribute changes, for example, the TeMIP platform uses the Acanthis Notification Exporter FM to export their OSI state change events.

This avoids the export session polling lag because events are exported automatically as soon as they arrive. Nevertheless, this method’s behavior when bursts of events are received requires careful testing. The data’s granularity is at the managed entity level.

The study takes the standard TeMIP OSI events containing attribute values (status changes, attribute value changes, and relationship changes) as an example.

3.5.1 TeMIP OSI configuration events

This use case actually relates to the TeMIP OSI State Change event, TeMIP Attribute Value Change event and the TeMIP Relationship Change event in this event partition.

3.5.1.1 Definition

These events contain the following attributes.

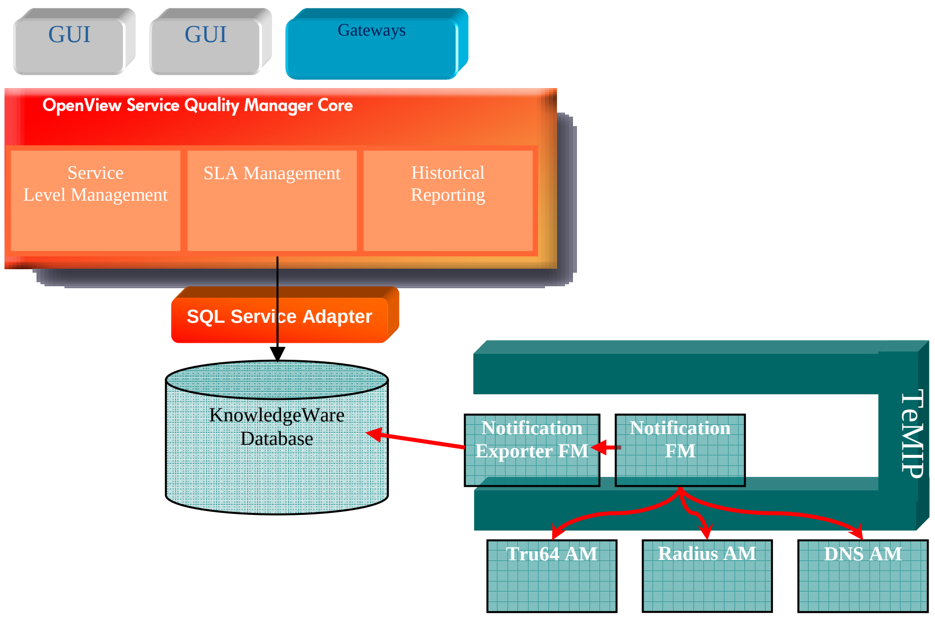
Name	Comment
Event Type	
Event Time	
Additional Text	
Source Indicator	
Attribute Identifier List	
State | Relationship | Attribute Value Change Definition	Contains the list of attributes, with their old and new values
Notification Identifier	
Correlated Notifications	
Additional Information	
Managed Object	

Only the *State Change Definition*, *Event Time* and the associated *Managed Object* are of interest in this use case.

3.5.1.2 Architecture

The Acanthis Notification Exporter FM is needed to export TeMIP events.

The following diagram illustrates how the Notification Exporting FM is used to integrate TeMIP and SQM.



Unfortunately, very few management modules systematically issue these events. In the TeMIP state management context, the State Mapper FM provides a good example of a module that issues these events when a generic state changes. Every managed entity supports these events and the *GetEvent* directive.

3.5.2 DFD definition

The DFD is identical to that of TeMIP Generic States. Only its behavior and extraction method are different.

Chapter 4

Integration lifecycle

This chapter describes each successive step you must perform when you integrate TeMIP and SQM in a working solution, using the Acanthis products and the SQL SA toolkit. It contains appropriate examples illustrating the configuration and adaptation required in each step.

Its workflow details only the sequence of steps involved in implementing this process. You do not need to implement each element contained in this chapter of the Implementation Cookbook, because the various elements it contains relate to different implementation situations.

4.1 Configuring TeMIP

If the appropriate Access Module and Functional Module have already been installed on the TeMIP directors, you need only configure TeMIP to prepare the collection domains used to collect exported alarms and events.

The following examples are based on a TNS directory created by entering the following command on the TeMIP director:

```
> tnscp

tnscp> create directory sqm
```

As in any TeMIP configuration, the network operators must then define the operation context overlying their collection domains to monitor alarms occurring on the entities they manage.

4.1.1 Collection domains

Collection domains are used to collect alarms or events on the entities they contain. They are used by the Alarm Handling FM and the Notification FM.

You can discover all managed entities they contain that can issue events or alarms (these are collected by the operator’s operation context) by listing all domain members and only discovering TeMIP global instances within the domain.

Unfortunately, this method provides insufficiently detailed information. You must then use the TeMIP dictionary to browse the child classes and lastly execute wildcard *show* requests for each global entity and its child entities.

An alternative solution consists in listing them directly and explicitly as elements of the domain. (This solution is recommended for TeMIP V5.X; it is not possible on the TeMIP V4.X platform.)

On TeMIP V4 platforms, managed entities are grouped into domains, as domain members. This means that the domain contains global entities, because members can only be used to track global entities.

The following example shows a collection domain created using the FCL PM. It is contains members that include a subdomain and a global instance. The subdomain might itself contain entities whose alarms TeMIP must collect.

```
> manage

TeMIP> create DOMAIN local_ns:.sqm.alarms_statistics_domain

TeMIP> create DOMAIN local_ns:.sqm.alarms_statistics_domain –
              MEMBER local_ns:subdomain

TeMIP> create DOMAIN local_ns:.sqm.alarms_statistics_domain –
              MEMBER local_ns:globalclass_instance
```

On TeMIP V5 platforms, managed entities are grouped into domains, as either members or elements. HP recommends you create domain elements, because these can be used to handle any type of entity, and their associated members are automatically created and deleted for each element as necessary. This means that domains can contain either global, child or wildcard entities that you can explicitly list as belonging to the domain.

The following example shows a collection domain created using the FCL PM. It contains elements that include a subdomain, global instance, child instance, and wildcard instance. The subdomain might itself contain entities whose events TeMIP must collect.

```
> manage

TeMIP> create DOMAIN local_ns:.sqm.events_aggregation_domain

TeMIP> create DOMAIN local_ns:.sqm.events_aggregation_domain –
              ELEMENT "subdomain" –
              Related Entity = Domain local_ns:.subdomain

TeMIP> create DOMAIN local_ns:.sqm.events_aggregation_domain –
              ELEMENT "globalinstance" -
              Related Entity = OSI_SYSTEM -
                                 local_ns:.directorname_local

TeMIP> create DOMAIN local_ns:.sqm.events_aggregation_domain -
              ELEMENT "childinstance" -
              Related Entity = OSI_SYSTEM -
                                 local_ns:.directorname_local –
                               TESTOBJ test_local

TeMIP> create DOMAIN local_ns:.sqm.events_aggregation_domain -
              ELEMENT "wildinstance" -
              Related Entity = BSS bss1 CARD *
```

The Alarm Handling FM associates each collection domain with an operation context. The Extended Archiving FM then uses the collection domains by associating each operation context with an extended archiving context. The Notification Exporter also uses FM Collection domains to export events collected on the domain, however.

4.1.2 Operation contexts

Operation contexts represent the network operator's view of the alarms occurring on a set of entities. It provides all the services needed to manage the alarm lifecycle.

Each operation context must have an associated collection domain. The collection domain's entities (DFIs) are those whose alarms must be exported to SQM. If not, TeMIP exports alarms on entities that are not required in calculating the SQL view.

Operation contexts are used to do the following:

- Filter and aggregate alarms
- Export associated attributes
- Calculate statistics on an alarm set
- Provide support for independent discovery processes

The scope of a specific context is not necessarily suited to the SQM fault statistic requirements.

If you are integrating an SQM platform with an existing TeMIP platform, the operation contexts may already be defined. If the existing operation contexts have a broader scope, you must take this into account when you perform the integration. In such cases, you must adapt the integration to avoid calculating statistics for all entities in the operation context's domain if only a subset is of use to the SQM operator.

Where possible, it is better to define collection domains specifically for the SQM integration's use case, possibly on a dedicated TeMIP director. All entities used in calculating fault statistics are then grouped into these domains. The operation contexts are fully dedicated to feeding the SQM platform with data.

Both approaches are valid, and you must take them both into consideration when you design the final integrated solution.

The following example shows an operation context created on top of the collection domain created previously for calculating alarm statistics.

```
> manage

TeMIP> create OPERATION_CONTEXT -
                    local_ns:.sqm.alarms_statistics_oper -
              Associated Domain = -
                    local_ns:.sqm.alarms_statistics_domain
```

4.2 Configuring Acanthis

Configuring the Acanthis products is an important step in the integration process, because this defines exactly which TeMIP data must be exported into the Oracle database.

Once you have installed the related Acanthis Data Exporter Functional Modules and configured, mounted and started the database to which they are linked, you must define exactly which data must be exported.

You must set up the database in several steps that must be performed in a specific order. You can choose to distribute the instances on several servers if necessary, or share them on a single server.

Database configuration is effectively a two-stage process consisting in:

1. Creating and customizing the database's specific tables;
2. Creating the contexts needed to export the data.

The configuration required varies considerably depending on which Data Exporter Functional Module is used. It respects this global approach in every case, however.

The following Acanthis Data Exporter Functional Modules are available:

- Attribute Exporter FM
- Notification Exporter FM
- Extended Archiving FM

The rest of this chapter contains additional details and examples of this approach. For further details of the process, see the related Acanthis documentation.

4.2.1 Setting up the databases

You can install the Oracle server locally or remotely depending on the specific environment and requirements. The only configuration requirement is that the TeMIP director must be able to access these Oracle database instances.

The Acanthis database instances are:

Instance name	TNS names	Table space	User Password
KNWA	*KNWASTAG*	*Acanthis*	`acanthis/acanthis`
		Temipaharchi	`temipaharchi/temipaharchi`
			`staging/statistics`
	KNWAPROD	*Production*	`production/statistics`
		Bo	`bo/`
KNWAARCH	*KNWAARCH*	*Archive*	`archive/statistics`

The Acanthis database that requires most configuring when you are integrating the platforms is the *KNWASTAG* instance, because it receives the exported TeMIP raw data. Each of these databases is OFA compliant.

Before you integrate the platforms, you must mount and start the database instances. You must then start the listener on the Oracle server and start these instances' services. (You can check `/etc/oratab` to see which databases must be started manually using the *dbstart* command or when the integrated system is started, and check *listener.ora* in `$ORACLE_HOME/network/admin` to find a client that can access the databases.)

You must also check that the `$TNS_ADMIN/tnsnames.ora` file is up-to-date on the local TeMIP director, so that the *sqlplus* tool can connect to the database instances remotely. You can do so by checking that *TNS_ADMIN* = `/var/opt/temip/conf`.

You can either install and configure the Acanthis databases separately, or group them together in a single database if they are installed as an Acanthis KnowledgeWare solution. You can also set them up on different database servers if necessary. For further details of how to set up Acanthis databases, see the *ADB Tool User's Guide* in the Acanthis documentation set.

HP recommends that you use the *adbtool* script to create and to configure all the Acanthis databases as a single database automatically, however. This is because you must set up the database by performing various steps in the correct order due to internal dependencies between the tables.

Even if all the integrated solution does not require all the tables, because it does not use all the exporting functions, you set up a consistent environment and platform that can evolve over time as necessary by using the *adbtool* script to create and to configure all the Acanthis databases as a single database.

If the integrated solution only requires one exporting module, you should of course implement the simplest and lightest solution.

4.2.2 Exporting attributes

The Attribute Exporter FM exports attribute values according to **attribute export contexts** and **export rules** that control how the attribute values of TeMIP Network Elements are exported to the database. You must configure these attribute export contexts and export rules for your specific environment.

4.2.2.1 Configuring the database tables

The Attribute Exporter FM works with the *KNWASTAG* database.

You must specify export rules that define which attributes the Attribute Exporter FM must export and how these must be exported. You do so using the OV TeMIP dictionary-driven `oracle_aefm_configuration_tool` utility.

This utility produces both the export rules and the appropriate SQL scripts (for the Oracle version only) used to create the tables that store attribute values.

The following example shows how to create the export rules for the TeMIP Generic States partition of the TeMIP IST Tru64 compaq cpqHostOs cpqHoComponent cpqHoUtil cpqHoCpuUtilTable class. For further details of how to create these export rules, see UC1.

```
> oracle_aefm_configuration_tool

***************************************************************
*                                                             *
*                                                             *
*                ORACLE  ATTRIBUTE  EXPORTING  FM             *
*                                                             *
*                      CONFIGURATION  TOOL                    *
*                                                             *
*                                                             *
***************************************************************

===================        Class ID        ===================

Enter class ID with following format:
   <global entity class ID>.<sub entity class ID>.<sub entity c
lass ID>

Enter class ID: 3000.232.11.2.3.1

Do you really want to export the Tru64.compaq.cpqHostOs.cpqHoCo
mponent.cpqHoUtil.cpqHoCpuUtilTable class [YES or NO] (default:
 YES):

=============       General rule description       ============

Tru64.compaq.cpqHostOs.cpqHoComponent.cpqHoUtil.cpqHoCpuUtilTab
le rule:

  * General characteristics [T|C|E]:
      T  | Target                      | KNWASTAG
      C  | Control character mapping   | space
      E  | Entity length               | 255
  * Partition list [ID]:
      1  | Identifiers
      4  | Characteristics
      59 | Generic State

  * Generate rule file [G].

Enter your choice: 59

==============       Partition description       ==============

Characteristics partition:

  * Attribute list [ID|'D'ID]:
      745356 | Generic Composite Operational State       |
BIDT_ENUMERATION | varchar2 | 255
      745357 | Generic Refresh Time Stamp                |
BinAbsTim        | date     | 0
      745358 | Generic Global Change Time Stamp          |
BinAbsTim        | date     | 0
      745359 | Generic Managed Status                    |
Boolean          | varchar2 | 255
      745360 | Generic Managed Status Change Time Stamp |
BinAbsTim        | date     | 0
      745361 | Generic Testing Status                    |
Boolean          | varchar2 | 255
      745362 | Generic Testing Status Change Time Stamp |
BinAbsTim        | date     | 0
      745363 | Generic Unknown Status                    |
Boolean          | varchar2 | 255
      745364 | Generic Operational State                 |
BIDT_ENUMERATION | varchar2 | 255
      745365 | Generic Usage State                       |
BIDT_ENUMERATION | varchar2 | 255
      745366 | Generic Alarm Status                      |
BIDT_ENUMERATION | varchar2 | 255
      745367 | Generic Availability Status               |
BIDT_ENUMERATION | varchar2 | 255
      745368 | Generic Administrative State              |
BIDT_ENUMERATION | varchar2 | 255
      745369 | Generic Procedural Status                 |
BIDT_ENUMERATION | varchar2 | 255
      745370 | Generic Control Status                    |
BIDT_ENUMERATION | varchar2 | 255
      745371 | Generic Standby Status                    |
BIDT_ENUMERATION | varchar2 | 255

  * Export all attributes with default values [A].

  * Return to general rule description [R].

Enter your choice: A

============        General rule description        ===========

Tru64.compaq.cpqHostOs.cpqHoComponent.cpqHoUtil.cpqHoCpuUtilTab
le rule:

  * General characteristics [T|C|E]:
      T  | Target                      | temip_ae
      C  | Control character mapping | space
      E  | Entity length               | 255
  * Partition list [ID]:
      1  | Identifiers
      4  | Characteristics
     >59 | Generic State

  * Generate rule file [G].

Enter your choice: G
Do you really want to generate a rule file [YES or NO] (default
: YES):

***************************************************************
*                                                             *
*                                                             *
*              ORACLE  ATTRIBUTE  EXPORTING  FM               *
*                    CONFIGURATION  TOOL                      *
*                                                             *
*                     FILE(S) GENERATED                       *
*                                                             *
*                                                             *
***************************************************************

oracle_aefm_temip_ae_Tru64_compaq_cpqHostOs_cpqHoComponent_cpqH
oUtil_cpqHoCpuUtilTable.cfg file generated.
oracle_aefm_temip_ae_Tru64_compaq_cpqHostOs_cpqHoComponent_cpqH
oUtil_cpqHoCpuUtilTable.csh file generated.
```

You must copy the configuration file under the directory `/var/opt/temip/awh/aefm/oracle/cfg` with read rights for the TeMIP user.

You must execute the shell script on the Oracle server to create the tables for the Generic States attribute values' export context for the selected class.

Alternatively, you can use the configuration tool to select which attributes must be exported if not the entire partition is not required. By default, each attribute is mapped to a specific Oracle data type. Fortunately, you can tune these by selecting an alternative data type if the default type does not meet your requirement in the integrated solution.

You must carefully configure the details of each class attribute in turn.

4.2.2.2 Configuring the export context

The Attribute Exporter FM's module manages attribute export contexts that ensure the export workflow works properly.

The attribute export context defines how the TeMIP attributes of the TeMIP entities belonging to the context (the **context members**) are exported. Context members are created in the same way as domain members.

The following example shows how to create the attribute export context for the TeMIP Generic States partition of the TeMIP IST Tru64 compaq cpqHostOs cpqHoComponent cpqHoUtil cpqHoCpuUtilTable class. For further details of this example, see UC1.

```
> manage

TeMIP> register ORACLE_AEC .sqm.aec_tru64_genstate –
                    Operation = plan, -
                    Managing Director = -
                           .temip.directorname_director
```

```
TeMIP> create ORACLE_AEC .sqm.aec_tru64_genstate –
                    Description = ORACLE_AEC Tru64 example, -
                    Target = KNWASTAG, -
                    Rows Commit Number = 1, -
                    Exporting Mode = Insert, -
                    Begin Time = now, -
                    Polling Period = 00:00:15, -
                    Export Area Management Mode = Delete, -
                    Export Area Management Age = 24:00:00, -
                    Maximum Number Of Exporting Threads = 10

TeMIP> register ORACLE_AEC .sqm.aec_tru64_genstate

TeMIP> create ORACLE_AEC .sqm.aec_tru64_genstate –

 MEMBER "Tru64 systemname Compaq cpqHostOs cpqHoComponent cpqHoU
til cpqHoCpuUtilTable"
```

Once you have created the attribute export context, you can easily add new members to it if the scheduling and export policy defined in the context matches their requirements.

HP recommends you define an attribute export context for each group of entities with identical scheduling and export policies. Alternatively, you can define one context for each entity class.

Attribute export contexts are managed in the same way as operation contexts. You can use standard directives such as *suspend*, *resume*, *activate*, *show* and *resetcounters* to control an attribute export context, show its attributes and check whether it is working properly.

```
> manage

TeMIP> show ORACLE_AEC .sqm.aec_tru64_genstate all attributes
```

The *status* partition is useful for checking the status of the data collection and export processes, while the *counters* partition is useful for displaying quantitative data to validate process progress.

4.2.2.3 Processing the export context

The attribute export context performs a *show* directive for all entities listed as context members, for all partitions needed in retrieving the list of attributes for export to the database periodically or on request. It then converts these attribute values into data type values and stores them in the database, using a separate Functional Module, the Oracle Data Exporter FM. If an export session takes longer than the polling interval, the integrated system simply skips the next export session.

The Attribute Exporter FM used (the Oracle version) offers configurable recovery capabilities enabling the integrated system to recover when an export fails because the Oracle database cannot be accessed.

The data collection and translation process is therefore separate from the database export process, meaning that no data is lost if the integrated system cannot access the database. If the database connection is lost or cannot be established, or if the Oracle Data Exporter FM crashes, the Attribute Exporter FM continues to create command files. The Oracle Data Exporter FM then processes these files the next time it connects to the database successfully. (It tries to connect to the database at the beginning and end of each collection session.)

4.2.2.4 Configuring the purge mechanism

You must set up a purge mechanism for each export context so that the database does not gradually become full.

Potential purge strategies vary considerably depending on the data volume and exporting rate concerned. Each of the following solutions is suited to a specific solution size.

As already said, you can configure the integrated solution so that the attribute export context exports data in one of two ways. By selecting Update export mode, you can ensure that if the integrated solution manages a limited number of entities, the data tables will never contain more than one row per entity. If you select Insert export mode, you must purge the export tables, however.

One database strategy consists in deleting from the table some rows for a managed entity each time data is inserted into the table for that entity. This requires a trigger mechanism that applies a predefined purge policy, such as keeping a managed entity’s attribute values if a specific attribute it contains equals a certain value. Such purge mechanisms can be very costly in terms of system resources, and they are not recommended for large volumes of data.

Another approach consists in executing a post-export script automatically after each export session, before the next export session begins. This script can use a PL/SQL purge function to easily maintain the number of rows of data in the table by applying a predefined purge policy (such as purging data that is more than 24 hours old, for example). Purging data after each export session can also be excessively costly in terms of system resources if the integrated solution must handle large volumes of data, but may be suitable for medium-sized solutions.

Alternatively, if it takes too long to purge data after each exporting session (the second approach above), you can set up an additional export context that is executed periodically so that its post-export script cleans up any tables as appropriate. This export context does not export any data and can be completely desynchronized from true export contexts.

4.2.3 Exporting notifications

The Notification Exporter FM exports event data according to **notification export contexts** and **export rules** that control how the event data of TeMIP Network Elements are exported to the database. You must configure these notification export contexts and export rules for your specific environment.

4.2.3.1 Configuring the database tables

The Notification Exporter FM works with the same *KNWASTAG* database used when the platform exports attributes.

You must specify export rules that define which events the Notification Exporter FM must export and how these must be exported. You do so using the OV TeMIP dictionary-driven `oracle_nefm_configuration_tool` utility.

This OV TeMIP utility produces both the data export rules and the appropriate SQL scripts (for the Oracle version only) used to create the tables that store the event attribute values.

The following example shows how to create the export rules for the TeMIP OSI State Change Event. For further details of this example, see UC2.

```
> oracle_nefm_configuration_tool

***************************************************************
*                                                             *
*                                                               *
*              ORACLE  NOTIFICATION  EXPORTING  FM              *
*                                                               *
*                      CONFIGURATION  TOOL                      *
*                                                               *
*                                                               *
*****************************************************************


================            Class ID            ===============

Enter class ID with following format:
        <global entity class ID>.<sub entity class ID>.<sub ent
ity class ID>

Enter class ID: 410
Do you really want to export the TEMIP class [YES or NO] (defau
lt: YES):


============        General rule description        ===========

TEMIP rule:

  * General characteristics [T|C|E|S|R|Q]:
      T    | Target                            | KNWASTAG
      C    | Control character mapping         | space
      E    | Entity length                     | 255
      S    | Exporting session name length     | 255
      R    | Generic rule                      | no
      Q    | SQL table name                    | NULL

  * Event list [ID|'D'ID]:
      1702 | Communications Alarm
      1703 | Environmental Alarm
      1704 | Equipment Alarm
      1705 | Integrity Violation
      1708 | Operational Violation
      1709 | Physical Violation
      1710 | Processing Error Alarm
      1711 | Quality Of Service Alarm
      1713 | Security Service Or Mechanism Violation
      1715 | Time Domain Violation
      1801 | Attribute Value Change
      1806 | Object Creation
      1807 | Object Deletion
      1812 | Relationship Change
      1814 | State Change

  * Generate rule file [G].

Enter your choice: R

===============          Generic rule          ==============

Generic rule:

        1 | yes
        2 | no

Enter generic rule (default: 2): 1
```

```
==========        General rule description       ============

TEMIP rule:

  * General characteristics [T|C|E|S|R|Q]:
      T    | Target                             | KNWASTAG
      C    | Control character mapping          | space
      E    | Entity length                      | 255
      S    | Exporting session name length      | 255
      R    | Generic rule                       | yes
      Q    | SQL table name                     | NULL

  * Event list [ID|'D'ID]:
      1702 | Communications Alarm
      1703 | Environmental Alarm
      1704 | Equipment Alarm
      1705 | Integrity Violation
      1708 | Operational Violation
      1709 | Physical Violation
      1710 | Processing Error Alarm
      1711 | Quality Of Service Alarm
      1713 | Security Service Or Mechanism Violation
      1715 | Time Domain Violation
      1801 | Attribute Value Change
      1806 | Object Creation
      1807 | Object Deletion
      1812 | Relationship Change
      1814 | State Change

  * Generate rule file [G].

Enter your choice: Q

=============           SQL Table Name          ==============

Enter the SQL table name [<= 30 char] (default: NULL): OSI_STAT
E_CHANGE

=============      General rule description     ==============

TEMIP rule:

  * General characteristics [T|C|E|S|R|Q]:
      T    | Target                             | KNWASTAG
      C    | Control character mapping          | space
      E    | Entity length                      | 255
      S    | Exporting session name length      | 255
      R    | Generic rule                       | yes
      Q    | SQL table name                     |
OSI_STATE_CHANGE

  * Event list [ID|'D'ID]:
      1702 | Communications Alarm
      1703 | Environmental Alarm
      1704 | Equipment Alarm
      1705 | Integrity Violation
      1708 | Operational Violation
      1709 | Physical Violation
      1710 | Processing Error Alarm
      1711 | Quality Of Service Alarm
      1713 | Security Service Or Mechanism Violation
      1715 | Time Domain Violation
      1801 | Attribute Value Change
      1806 | Object Creation
      1807 | Object Deletion
      1812 | Relationship Change
      1814 | State Change

  * Generate rule file [G].

Enter your choice: 1814


=============         Event description         ==============

State Change Event:

  * Argument list [ID|'D'ID]:
      9   | Managed Object             | FullEntityName    |
varchar2 | 255
      1   | Event Type                 | BIDT_ENUMERATION |
varchar2 | 255
      2   | Event Time                 | BinAbsTim         | date
| 0
      15 | Source Indicator           | BIDT_ENUMERATION |
varchar2 | 255
      13 | Attribute Identifier List | BIDT_SETOF        |
varchar2 | 255
      14 | State Change Definition    | BIDT_SETOF        |
varchar2 | 255
      5   | Notification Identifier    | Unsigned32        |
number   | 38
      6   | Correlated Notifications   | BIDT_SETOF        |
varchar2 | 255
      7   | Additional Text            | Latin1String      |
varchar2 | 255
      8   | Additional Information     | BIDT_SETOF        |
varchar2 | 255

  * Export all arguments with default values [A].

  * Return to general rule description [R].

Enter your choice: A

============        General rule description       ============

TEMIP rule:

  * General characteristics [T|C|E|S|R|Q]:
      T     | Target                             | KNWASTAG
      C     | Control character mapping          | space
      E     | Entity length                      | 255
      S     | Exporting session name length      | 255
      R     | Generic rule                       | yes
      Q     | SQL table name                     | State_Change

  * Event list [ID|'D'ID]:
      1702 | Communications Alarm
      1703 | Environmental Alarm
      1704 | Equipment Alarm
      1705 | Integrity Violation
      1708 | Operational Violation
      1709 | Physical Violation
      1710 | Processing Error Alarm
      1711 | Quality Of Service Alarm
      1713 | Security Service Or Mechanism Violation
      1715 | Time Domain Violation
      1801 | Attribute Value Change
      1806 | Object Creation
      1807 | Object Deletion
      1812 | Relationship Change
     >1814 | State Change

   * Generate rule file [G].

Enter your choice: G

Do you really want to generate a rule file [YES or NO] (default
: YES):

****************************************************************
*                                                              *
*                                                              *
*             ORACLE  NOTIFICATION  EXPORTING  FM              *
*                     CONFIGURATION  TOOL                      *
*                      FILE(S) GENERATED                       *
*                                                              *
****************************************************************

oracle_nefm_temip_ne_NEFMGENERIC_State_Change.cfg file generate
d.
oracle_nefm_temip_ne_NEFMGENERIC_State_Change.csh file generate
d.
```

You must copy the configuration file under the /`var/opt/temip/awh/nefm/oracle/cfg` directory with read rights for the TeMIP user.

You must execute the shell script on the Oracle server to create the tables for the state change events' export context for the selected class.

In this example, a generic rule is generated, meaning the generated table is valid for any TeMIP class and all state change events are collected, regardless of the managed entity's class.

Alternatively, you can generate an export rule for a specific class if necessary, and you can choose to export only a subset of the event's attribute data.

With the exception of the generic rule use case, the configuration process is similar to that used to configure the attribute export process.

4.2.3.2 Configuring the export context

The Notification Exporter FM manages notification export contexts that ensure the export workflow works properly.

The notification export context defines the TeMIP events of the TeMIP entities belonging to the collection domain concerned are exported.

The following example shows how to create the notification export context for the *TeMIP OSI State Change* event of entities in the *sqm.events_aggregation_domain* domain. For further details of this example, see UC2.

```
> manage

TeMIP> register ORACLE_NEC .sqm.nec_statechange –
                    Operation = plan, -
                    Managing Director = -
                           .temip.directorname_director
```

```
TeMIP> create ORACLE_NEC .sqm.nec_statechange –
                    Description = ORACLE_NEC example, -
                    Target = KNWASTAG, -
                    Rows Commit Number = 1, -
                    Associated Domain = Domain –
                     local_ns:.sqm.events_aggregation_domain, -
                    Discriminator Construct = {}, -
                    Notify Entity List = {}, -
                    Notify Events = -
                     ( Any Configuration Events ), -
                    Exporting Session Name = -
                     "ORACLE_NEC Tru64 example", -
                    Maximum Number Of Exporting Threads = 10

TeMIP> register ORACLE_NEC .sqm.nec_statechange
```

Once you have created the notification export context, you must select which entities must be managed, and export events by modifying the associated domain’s content.

HP recommends you define a domain for each group of managed entities with identical associated events and filtering capabilities. Alternatively, you can define a context for each domain, containing only entities that are scoped under a global class.

Notification export contexts are managed in the same way as operation contexts. You can use standard directives such as *suspend*, *resume*, *show* and *resetcounters* to control a notification export context, show its attributes and check whether it is working properly.

```
> manage

TeMIP> show ORACLE_NEC .sqm.nec_statechange all attributes
```

The *status* partition is useful for checking the status of the data collection and export processes, while the *counters* partition is useful for displaying quantitative data to validate process progress.

4.2.3.3 Processing the export context

When it is activated, the notification export context listens to any events occurring on managed entities in its domain. When it does so, the platform applies the following filters, in order of importance:

- Event list, defining which events must be listened to
- Entity list, defining which entities are of interest
- Discriminator construct, offering more advanced filtering possibilities

The platform then exports filtered events after decoding them and converting them to the appropriate data type format.

Notification export contexts do not include any concept of start time or periods, because once they are activated they collect all filtered events and export them immediately to the database.

4.2.3.4 Configuring the purge mechanism

You must set up a purge mechanism for each export context so that the database does not gradually become full.

Potential purge strategies vary considerably depending on the data volume and exporting rate concerned. Each of the following solutions is suited to a specific solution size.

One database strategy consists in deleting from the table some rows for a managed entity each time data is inserted into the table for that entity. This requires a trigger mechanism that applies a predefined purge policy, such as keeping a managed entity's attribute values if a specific attribute it contains equals a certain value. Such purge mechanisms can be very costly in terms of system resources, and they are not recommended for large volumes of data.

Another approach consists in executing a post-export script automatically after each export session, before the next export session begins. This script can use a PL/SQL purge function to easily maintain the number of rows of data in the table by applying a predefined purge policy (such as purging data that is more than 24 hours old, for example). Purging data after each export session can also be excessively costly in terms of system resources if the integrated solution must handle large volumes of data, but may be suitable for medium-sized solutions.

Alternatively, if it takes too long to purge data after each exporting session (the second approach above), you can set up an additional export context that is executed periodically so that its post-export script cleans up any tables as appropriate. This export context does not export any data and can be completely desynchronized from true export contexts.

4.2.4 Extended archiving

The Extended Archiving FM exports alarm content according to **extended archiving contexts** that control how TeMIP Network Element alarms are exported to the database. You must configure these extended archiving contexts for your specific environment.

4.2.4.1 Configuring the database tables

You do not need to configure the Oracle database because the necessary environment (such as the user, tablespace, tables, and PL/SQL functions) is configured automatically when the database is created.

4.2.4.2 Configuring the export context

The only configuration required consists in defining which operation contexts must be used in an extended archiving context. You can add new operation contexts to the extended archiving context if they have a similar export policy; otherwise, you must add them to a different extended archiving context. For further details of this example, see UC3.

```
> manage

TeMIP> register ORACLE_EAC .sqm.eac_alarms_statistics –
                    Operation = plan, -
                    Managing Director = -
                           .temip.directorname_director

TeMIP> create ORACLE_EAC .sqm.eac_alarms_statistics –
       Description = "ORACLE_EAC Alarms Statistics example"

TeMIP> register ORACLE_EAC .sqm.eac_statechange

TeMIP> create ORACLE_EAC .sqm.eac_alarms_statistics –
               MEMBER -
       "OPERATION_CONTEXT local_ns:.sqm.alarms_statistics_oper"
```

Once you have created the extended archiving context, you can add the operation contexts to the extended archiving context.

HP recommends you have one extended archiving context for each set of operation contexts under the responsibility of a specific operator.

Extended archiving contexts are managed in the same way as operation contexts. You can use standard directives such as *suspend*, *resume*, *activate*, *show*, and *resetcounters* to control it, show its attributes and check whether it is working properly.

```
> manage

TeMIP> show ORACLE_EAC .sqm.eac_alarms_statistics all
attributes
```

The *status* partition is useful for checking the status of the data collection and export processes, while the *counters* partition is useful for displaying quantitative data to validate process progress.

4.2.4.3 Processing the export context

An extended archiving context handles a list of operation contexts that belong to the archiving context.

Before the platform begins processing and storing previously unarchived alarm objects, it deletes all rows linked to previous archiving sessions, to avoid database inconsistencies. This ensures that rows stored using the *archive* directive are never modified or deleted.

The platform then reads all unarchived alarm objects, translates them into an exportable format, and stores them in the same Oracle database and SQL tables as when the standard OV TeMIP archiving function is used.

4.2.4.4 Configuring the purge mechanism

You must set up a purge mechanism for each export context so that the database does not gradually become full.

Potential purge strategies vary considerably depending on the data volume and exporting rate concerned. Each of the following solutions is suited to a specific solution size.

The extended archiving context is provided with a set of attributes for tuning the alarm export and purge strategy. For further details of how to use the Extended Archiving FM, see the associated documentation.

One database strategy consists in deleting from the table some rows for a managed entity each time data is inserted into the table for that entity. This requires a trigger mechanism that applies a predefined purge policy, such as keeping a managed entity’s alarm if a specific alarm attribute it contains equals a certain value. Such purge mechanisms can be very costly in terms of system resources, and they are not recommended for large volumes of data.

Another approach consists in executing a post-export script automatically after each export session, before the next export session begins. This script can use a PL/SQL purge function to easily maintain the number of rows of data in the table by applying a predefined purge policy (such as purging data that is more than 24 hours old, for example).

Alternatively, if it takes too long to purge data after each exporting session (the second approach above), you can set up an additional export context that is executed periodically so that its post-export script cleans up any tables as appropriate. This export context does not export any data and can be completely desynchronized from true export contexts.

4.3 Oracle views and functions

You may be able to work directly with the Acanthis database’s export tables to extract the parameters for passing to the SQM platform. In general, however, you must set up views, functions, triggers, and/or temporary tables to handle the additional processing needed to compute and format the data, and align it with SQM.

4.3.1 Temporary tables

Temporary tables are usually used in combination with triggers or stored PL/SQL procedures to process the exported data. You can also use them to drive function processing so that only a part of the exported data is used to compute certain added-value parameters.

The following example shows how to create a temporary table containing the result of an independent discovery process that is used as a filter on the raw data and is used in a PL/SQL script to drive its processing. For further details of this example, see UC3.

```
CREATE TABLE MANAGEDOBJECTS(
        Entity_Name varchar2(512) not null,
        Domain_Name varchar2(512) not null,
        Operation_Context varchar2(512) not null,
        Scope varchar2(20) not null
);
```

The following example shows how to create a temporary table that contains only the transitional attribute values of an entity that changes over time (see UC1):

```
CREATE TABLE tmp_tru64 (
        Entity_Name varchar2(512) not null,
        Timestamp date,
        Generic_Composit_Operati_State varchar2(255),
        Generic_Refresh_Time_Stamp date,
        Generi_Global_Chang_Time_Stamp date,
        Generic_Managed_Status varchar2(255),
        Gener_Mana_Stat_Chan_Time_Stam date,
        Generic_Testing_Status varchar2(255),
        Gener_Test_Stat_Chan_Time_Stam date,
        Generic_Unknown_Status varchar2(255),
        Generic_Operational_State varchar2(255),
        Generic_Usage_State varchar2(255),
        Generic_Alarm_Status varchar2(255),
        Generic_Availability_Status varchar2(255),
        Generic_Administrative_State varchar2(255),
        Generic_Procedural_Status varchar2(255),
        Generic_Control_Status varchar2(255),
        Generic_Standby_Status varchar2(255));
```

The following example shows to create two temporary tables. The first of these contains the event state change definition with several attribute value changes in it, while the second contains the attribute name and its value extracted from the state change definition, for each attribute in the state change definition (see UC5):

```
CREATE TABLE tmp_tru64_attr (entity_name varchar2(300), event_t
ime timestamp, state_change varchar2(255));

CREATE TABLE tmp_state_attr (entity_name varchar2(300), event_t
ime timestamp, attr_name varchar2(1024), attr_value varchar2(10
24));
```

4.3.2 Triggers

Triggers are quite costly in terms of system resource use but can be very useful in some cases, to simply update a temporary table or purge some data in another table automatically, for example.

They are used to populate temporary tables with relevant rows defined in their processing.

The following example shows how to update the temporary table containing attribute values only if at least one attribute value has changed between successive export sessions. For further details of this example, see UC1.

```
--
===============================================================
-- {{{ TRIGGER tru64_attr_update_trig
-- ------------------------------------------------------------
-- Trigger on attributes insertion
--
===============================================================
CREATE OR REPLACE TRIGGER tru64_attr_update_trig
    AFTER INSERT ON tru64 FOR EACH ROW
DECLARE
        entity varchar2(512);
        tstamp timestamp;
        grts varchar2(255);
        ggcts varchar2(255);
        gmscts varchar2(255);
        gtscts varchar2(255);
        gms varchar2(255);
        gts varchar2(255);
        gos varchar2(255);
        guns varchar2(255);
        gcos varchar2(255);
        guss varchar2(255);
        gals varchar2(255);
        gavs varchar2(255);
        gads varchar2(255);
        gps varchar2(255);
        gcs varchar2(255);
        gss varchar2(255);
        is_first_time boolean := FALSE;
BEGIN

        BEGIN

        select Entity_Name,
                Timestamp,
                Generic_Composit_Operati_State,
                Generic_Refresh_Time_Stamp,
                Generi_Global_Chang_Time_Stamp,
                Generic_Managed_Status,
                Gener_Mana_Stat_Chan_Time_Stam,
                Generic_Testing_Status,
                Gener_Test_Stat_Chan_Time_Stam,
                Generic_Unknown_Status,
                Generic_Operational_State,
                Generic_Usage_State,
                Generic_Alarm_Status,
                Generic_Availability_Status,
                Generic_Administrative_State,
                Generic_Procedural_Status,
                Generic_Control_Status,
                Generic_Standby_Status
          into entity, tstamp, gcos, grts, ggcts, gms, gmscts,
gts, gtscts, guns,
gos, guss, gals, gavs, gads, gps, gcs, gss
        from ( select Entity_Name,
                        Timestamp,
                        Generic_Composit_Operati_State,
                        Generic_Refresh_Time_Stamp,
                        Generi_Global_Chang_Time_Stamp,
                        Generic_Managed_Status,
                        Gener_Mana_Stat_Chan_Time_Stam,
                        Generic_Testing_Status,
                        Gener_Test_Stat_Chan_Time_Stam,
                        Generic_Unknown_Status,
                        Generic_Operational_State,
                        Generic_Usage_State,
                        Generic_Alarm_Status,
                        Generic_Availability_Status,
                        Generic_Administrative_State,
                        Generic_Procedural_Status,
                        Generic_Control_Status,
                        Generic_Standby_Status
                from tmp_tru64
                where tmp_tru64.Entity_Name = :new.Entity_Name
                order by Timestamp desc )
        where ROWNUM = 1;
        EXCEPTION WHEN NO_DATA_FOUND THEN
                is_first_time := TRUE;
        END;

        IF is_first_time
        THEN
             -- first time in final tmp_tru64 table
             insert into tmp_tru64
             values (:new.Entity_Name,
               :new.Timestamp,
               :new.Generic_Composit_Operati_State,
               :new.Generic_Refresh_Time_Stamp,
               :new.Generi_Global_Chang_Time_Stamp,
               :new.Generic_Managed_Status,
               :new.Gener_Mana_Stat_Chan_Time_Stam,
               :new.Generic_Testing_Status,
               :new.Gener_Test_Stat_Chan_Time_Stam,
               :new.Generic_Unknown_Status,
               :new.Generic_Operational_State,
               :new.Generic_Usage_State,
               :new.Generic_Alarm_Status,
               :new.Generic_Availability_Status,
               :new.Generic_Administrative_State,
               :new.Generic_Procedural_Status,
               :new.Generic_Control_Status,
               :new.Generic_Standby_Status);
        ELSE
             IF entity = :new.Entity_Name and (
               gcos != :new.Generic_Composit_Operati_State or
               grts != :new.Generic_Refresh_Time_Stamp or
               ggcts != :new.Generi_Global_Chang_Time_Stamp or
               gms != :new.Generic_Managed_Status or
               gmscts != :new.Gener_Mana_Stat_Chan_Time_Stam or
               gts != :new.Generic_Testing_Status or
```

```
                gtscts != :new.Gener_Test_Stat_Chan_Time_Stam or
                guns != :new.Generic_Unknown_Status or
                gos != :new.Generic_Operational_State or
                guss != :new.Generic_Usage_State or
                gals != :new.Generic_Alarm_Status or
                gavs != :new.Generic_Availability_Status or
                gads != :new.Generic_Administrative_State or
                gps != :new.Generic_Procedural_Status or
                gcs != :new.Generic_Control_Status or
                gss != :new.Generic_Standby_Status )
              THEN
                -- any change for same entity name
                insert into tmp_tru64
                values (:new.Entity_Name,
                                 :new.Timestamp,

:new.Generic_Composit_Operati_State,

:new.Generic_Refresh_Time_Stamp,

:new.Generi_Global_Chang_Time_Stamp,
                                 :new.Generic_Managed_Status,

:new.Gener_Mana_Stat_Chan_Time_Stam,
                                 :new.Generic_Testing_Status,

:new.Gener_Test_Stat_Chan_Time_Stam,
                                 :new.Generic_Unknown_Status,
                                 :new.Generic_Operational_State,
                                 :new.Generic_Usage_State,
                                 :new.Generic_Alarm_Status,

:new.Generic_Availability_Status,

:new.Generic_Administrative_State,
                                 :new.Generic_Procedural_Status,
                                 :new.Generic_Control_Status,
                                 :new.Generic_Standby_Status);

                END IF;
        END IF;
END tru64_attr_update_trig;
/
SHOW ERR;

-- }}}
```

The next example shows how to extract a state change definition field's attribute names and values, and then insert them in a temporary table containing the attribute name and the attribute value as generic columns. For further details of this example, see UC5.

```
--
===============================================================
-- {{{ TRIGGER notif_state_update_trig
-- -------------------------------------------------------------
-- Trigger on state attributes insertion
--
===============================================================
CREATE OR REPLACE TRIGGER notif_state_update_trig
    AFTER INSERT ON state_change FOR EACH ROW
    WHEN ( new.state_change_definition is not null )
DECLARE
        pos1_v number := 1;
        pos2_v number := 0;
        pos3_v number := 0;
        len_v number := 0;
        attr_name_v varchar2(255);
        attr_value_v varchar2(1024);
        state_change_v varchar2(4000);
BEGIN
        state_change_v := :new.state_change_definition;
        len_v := LENGTH(state_change_v);

        WHILE pos1_v < len_v
        LOOP
           -- Reach next newAttributeValue and position after e
qual sign
           pos1_v := INSTR(state_change_v, '=', pos1_v, 3);
           -- Reach associated first comma
           pos2_v := INSTR(state_change_v, ',', pos1_v, 1) + 2;
           -- Reach associated second comma
           pos3_v := INSTR(state_change_v, ',', pos2_v, 1) - 2;

           -- Get attribute name in between
           attr_name_v := SUBSTR(state_change_v, pos2_v, pos3_v
 - pos2_v);

           -- Reach next comma after attribute name
           pos2_v := pos3_v + 4;
           pos3_v := INSTR(state_change_v, ')', pos2_v, 1) - 1;

           -- Get attribute value in between
           attr_value_v := SUBSTR(state_change_v, pos2_v, pos3_
v - pos2_v);

           -- Prepare for next attribute
           IF (pos3_v + 4) > len_v THEN
                   -- Final exit
                   pos1_v := len_v + 1;
           ELSE
                   -- More attributes
                   pos1_v := pos3_v;
           END IF;

           --
 Insert attribute name/value retrieved from state_change_defini
tion
           insert into tmp_state_attr values (:new.entity_name,
 temip_intg_
tools.to_timestamp(:new.event_time), attr_name_v, attr_value_v)
;
        END LOOP;
END notif_state_update_trig;
/
SHOW ERR;

-- }}}
```

The next example shows the two-step process used to finally retrieve the list of monitored attributes for the newly-exported alarms. For further details of this example, see UC4.

```
--
================================================================
-- {{{ TRIGGER alarm_object_creation_trig
-- -------------------------------------------------------------
-- Trigger on alarm insertion.
--
================================================================
CREATE OR REPLACE TRIGGER alarm_object_creation_trig
    AFTER INSERT ON alarmobject0 FOR EACH ROW
DECLARE
        id_v number(22);
        is_already_known_v boolean := TRUE;
BEGIN
        BEGIN
           select identifier
           into id_v
           from tmp_alarm_object
           where tmp_alarm_object.identifier = :new.identifier;
        EXCEPTION WHEN NO_DATA_FOUND THEN
           is_already_known_v := FALSE;
        END;
        IF is_already_known_v = FALSE THEN
           insert into tmp_alarm_object values (:new.identifier
, :new.manag
edobject, :new.eventtime, :new.monitoredattributes);
        END IF;
END alarm_object_creation_trig;
/
SHOW ERR;

-- }}}


--
================================================================
-- {{{ TRIGGER mon_attr_creation_trig
-- -------------------------------------------------------------
-- Trigger on alarms monitoread attributes insertion.
--
================================================================
CREATE OR REPLACE TRIGGER mon_attr_creation_trig
    AFTER INSERT ON monitoredattributes FOR EACH ROW
DECLARE
        aoid_v number(22);
        mobj_v varchar2(300);
        tstamp_v timestamp;
        mattr_v number(22);
        nattr_v varchar2(1024);
        is_processed_v char;
        is_unknown_v boolean := FALSE;
        is_first_time_v boolean := FALSE;
        is_valid_v boolean := TRUE;
BEGIN
        BEGIN

        -- Check if monitored attribute is known

        select tmp_mon_attr.identifier,
                tmp_mon_attr.attr_name
        into mattr_v, nattr_v
        from      tmp_mon_attr
        right outer join
                ( select tmp_alarm_object.monattrid
                  from tmp_alarm_object
                  where tmp_alarm_object.monattrid = :new.monit
oredattributes
                ) t_alarm
        on t_alarm.monattrid = tmp_mon_attr.identifier
        where tmp_mon_attr.attr_name = temip_intg_tools.get_att
r_name(:new.monitoredattributes60708)
        and tmp_mon_attr.attr_value = temip_intg_tools.get_attr
_value(:new.monitoredattributes60708);

        EXCEPTION WHEN NO_DATA_FOUND THEN
                is_unknown_v := TRUE;
        END;

        IF is_unknown_v THEN

                BEGIN

                -- Retrieve missing fields from alarm

                select identifier,
                        managedobject,

temip_intg_tools.to_utc_timestamp(eventtime),
                        monattrid
                into aoid_v,
                        mobj_v,
                        tstamp_v,
                        mattr_v
                from tmp_alarm_object
                where monattrid = :new.monitoredattributes;

                EXCEPTION WHEN NO_DATA_FOUND THEN
                        is_valid_v := FALSE;
                END;

                -- Check if everything is available

                IF is_valid_v THEN

                        -- Update tmp_mon_attr table

                        insert
                        into tmp_mon_attr
                        values
                        (
                                mattr_v,
                                mobj_v,
                                tstamp_v,
                               temip_intg_tools.get_attr_name(:
new.monitoredattributes60708),
                               temip_intg_tools.get_attr_value(
:new.monitoredattributes60708)
                        );
                END IF;
        END IF;
END mon_attr_creation_trig;
/
SHOW ERR;

-- }}}
```

4.3.3 Added-value views

You can use these views to gather together data stored in several tables and perform processing based on certain column values. You can also use these views to define how the format of specified attributes must be translated to different formats. The SQL query involved can be very complex, and can include unions, joins, products, additional indexes, sorting, etc.

The following example shows how to create a view that obtains only a subset of attributes and converts a timestamp. For further details of this example, see UC1.

```
CREATE OR REPLACE VIEW TRU64_STATUS_VIEW as
(select ENTITY_NAME,
        GENERIC_OPERATIONAL_STATE,
        GENERIC_ADMINISTRATIVE_STATE,
        TEMIP_INTG_TOOLS.TO_UTC_TIMESTAMP(TIMESTAMP) as GMTTIME
 from TMP_TRU64);
```

The next example shows how to create a view that converts the received event’s type into a numerical value. For further details of this example, see UC2.

```
CREATE OR REPLACE VIEW IP_REACHABILITY_VIEW as
(
select
ENTITY_NAME, TEMIP_INTG_TOOLS.to_utc_timestamp(EVENT_TIME) as G
MTTIME, TEMIP_INTG_TOOLS.to_reachability(PERCEIVED_SEVERITY) as
 REACHABILITY
from ACANTHIS.IP_REACHABILITY_UP
union all
select ENTITY_NAME, TEMIP_INTG_TOOLS.to_utc_timestamp(EVENT_TIM
E) as GMTTIME, TEMIP_INTG_TOOLS.to_reachability(PERCEIVED_SEVER
ITY) as REACHABILITY
from ACANTHIS.IP_REACHABILITY_DOWN
);
```

The next example shows how to create a view that converts attribute names into column names, and then assigns them their attribute value as the column value. For further details of this example, see UC4.

```
CREATE OR REPLACE VIEW DNS_VIEW as
(
select t_grouped.managedobject, t_grouped.tstamp,
        t1.attr_value as dpServerAddress,
        t2.attr_value as dpSeverity,
        to_number(t3.attr_value) as dpResponseTime
from (select managedobject, tstamp from temipaharchi.tmp_mon_at
tr where substr(managedobject, 1, 3) = 'DNS' group by managedob
ject, tstamp) t_grouped
left outer join
     (select managedobject, tstamp, attr_value from temipaharch
i.tmp_mon_attr where attr_name = 'dpServerAddress ') t1
on t_grouped.managedobject = t1.managedobject and t_grouped.tst
amp = t1.tstamp
left outer join
     (select managedobject, tstamp, attr_value from temipaharch
i.tmp_mon_attr where attr_name = 'dpSeverity ') t2
on t_grouped.managedobject = t2.managedobject and t_grouped.tst
amp = t2.tstamp
left outer join
     (select managedobject, tstamp, attr_value from temipaharch
i.tmp_mon_attr where attr_name = 'dpResponseTime ') t3
on t_grouped.managedobject = t3.managedobject and t_grouped.tst
amp = t3.tstamp
);
```

The next example shows how to create a set of views that compute alarm statistics. To see a simple version that uses a limited list of statistic parameters, see UC3.

```
CREATE OR REPLACE VIEW STATISTICS_WORST_VIEW as
(
select managedobject,
        severity,
        eventtime,
        additionaltext
from
(
        select identifier,
                managedobject,

temip_intg_tools.string_to_severity(perceivedseverity) as sever
ity,
                eventtime,
                additionaltext,
                rank() over ( partition by managedobject order
by temip_intg_tools.string_to_severity(perceivedseverity), even
ttime, identifier) as rnk
        from temipaharchi.alarmobject0
        where state = 'Outstanding' and clearancetimestamp is N
ULL
)
where rnk = 1
);

CREATE OR REPLACE VIEW STATISTICS_COUNT_VIEW as
(
select distinct managedobject,
        (
                select count(*)
                from alarmobject0
                where state='Outstanding'
                        and clearancetimestamp is NULL
                        and managedobject= a0.managedobject
        ) as counter
from alarmobject0 a0
);

CREATE OR REPLACE VIEW STATISTICS_OLDEST_VIEW as
(
select managedobject,
        temip_intg_tools.minus_2AbsTime_sec(
                sys_extract_utc(systimestamp),

acanthis.temip_intg_tools.to_utc_timestamp(eventtime)) as durat
ion
from
(
        select identifier,
                managedobject,
                eventtime,
                rank() over ( partition by managedobject order
by eventtime, identifier) as rnk
        from temipaharchi.alarmobject0
        where state = 'Outstanding' and clearancetimestamp is N
ULL
)
where rnk = 1
);

CREATE OR REPLACE VIEW STATISTICS_VIEW as
(
select c.managedobject as managedobject,
        c.severity as severity,
        c.additionaltext as additionaltext,
        c.counter as counter,
        NVL(d.duration, '0') as duration,
        sys_extract_utc(systimestamp) as gmttime
from
(
        select b.managedobject as managedobject,
                NVL(a.severity, 0) as severity,
                NVL(a.additionaltext, 'No additional text avail
able') as additionaltext,
                b.counter as counter
        from STATISTICS_WORST_VIEW a
        right join STATISTICS_COUNT_VIEW b
        on a.managedobject = b.managedobject
) c
left join STATISTICS_OLDEST_VIEW d
on c.managedobject = d.managedobject
);
```

HP recommends you use a single view grouping into one row all parameters passed for a given managed entity, to simplify the data collection query in the SQL SA.

You may not be able to use views in all situations, however, and some additional processing may be required to obtain the correct values for passing to SQM. Such cases include situations such as passing the initial value, where no data is available, comparisons between the old value and the new value, unchanged data, data extraction and conversion into columns, among others. For further details of such additional processing, see chapter 4.3.2, "

Triggers", and chapter 4.3.4, "Stored procedures".

4.3.4 Stored procedures

Stored procedures are PL/SQL functions that are called when they are required to process table data. They are much more efficient than triggers, which are used systematically whenever tables are updated, and often work on each row in turn. Stored procedures are preloaded, and work on large datasets instead.

Their use in place of triggers is more a question of performance and volume than a question of feasibility.

4.4 SQL SA discovery

The SQL SA discovery feature is used to retrieve from TeMIP all instances that must be considered as DFIs within SQM. In general, this process is based on the entity name but it can also include other information, such as the domain name, customer name, etc.

The SQL SA is supplied with an automatic discovery mechanism that runs a SQL query on a table containing the list of managed entities. It offers a complete set of features, such as reference set handling and filtering.

All exported data includes an entity name, but if nothing is exported the associated table remains empty, meaning the discovery is performed unnecessarily. This may be suitable in some cases, but it is inappropriate in most cases.

Discovery must therefore be isolated from the exported data's availability. HP recommends you build a temporary table for the discovery process and use it to contain the list of discovered entities, and then use the default discovery mechanism supplied with the SQL SA. As an added advantage, when the managed entities are listed in a separate table, this table can also be used to drive parameter calculations with other tables.

The following example illustrates the discovery process performed to calculate the alarm statistics.

```
CREATE TABLE MANAGEDOBJECTS(
        Entity_Name varchar2(512) not null,
        Domain_Name varchar2(512) not null,
        Operation_Context varchar2(512) not null,
        Scope varchar2(20) not null
);
```

TeMIP discovery in this context is based on a tool that retrieves the list of a domain's elements recursively, and then inserts the associated entities in the temporary table *MANAGEDOBJECTS*. The discovery process is performed either each time the domain changes or periodically in a *crontab*.

```
> manage

TeMIP> getelement DOMAIN local_ns:.sqm.alarms_statistics_domain
TeMIP> getelement DOMAIN
local_ns:.sqm.alarms_statistics_domain_subdomain1
TeMIP> …
```

4.4.1 Granularity

A DFI's granularity depends on its MRP definition. In a TeMIP context, this normally simply consists in the entity name, which can always be found in exported data.

You may require additional information, however, if the entity belongs to several data collection or export context domains.

If the entity is in customer A's collection domain and is also in customer B's collection domain, for example, the domain name is needed to know whether the parameters were returned from collection in domain A or B. This customer information may also be retrieved into the *Subscriber* field.

HP also recommends you provide a script to filter all retrieved entities, as is done in the automatic SQL discovery process, before you load them into the final Oracle TeMIP entities table. You can then use this to transform the data or apply default rules, such as computing the scope (specific, wildcard, or wholetree) from the entity name.

4.4.2 Automated, or Manual

When you start from a new environment, you must initialize the table containing the list of TeMIP entities you want to manage as DFIs.

You must perform the SQL SA discovery process manually at least once. In addition, HP recommend that you test the integrated solution by beginning with a small domain and checking that everything is discovered and works correctly.

You can automate SQL SA discovery by running it in a *crontab* at appropriate intervals for the domain changes (daily, for example).

4.4.3 Filtering

The SQL SA discovery feature also includes the possibility of defining a filter to remove all entities that match regular expressions. For more information on regular expressions, see the *SQL SA Toolkit User's Guide*.

In this phase, the platform loads the discovery query definition file and runs the discovery query to retrieve all the SQL SA Data Feeder Instance definitions.

In the discovery filtering phase, the discovery tool executes a filtering script. This script parses the raw discovery file output by the previous command. It removes all DFI definitions that are not managed by the SQL SA application. This filtering is mainly used for load balancing (to share the DFI load over several SQL SA applications). The script generates a new DFI inventory file containing only DFIs that are managed by the SQL SA application.

A filtering script is supplied for customizing the SQL SA. This default script simply copies the input raw inventory file to the filtered inventory file without applying any filtering. You must adapt this script according to your needs.

Additionally, you can add to the SQL SA discovery query a *where* clause that specifies which of the discovered entities you want to select.

4.5 Updating the SQM model

The models defined in SQM serve both as service models and as models of the service level agreements linked to those service models. You may of course wish to show a certain amount of the TeMIP model within SQM.

SQM commonly sees the TeMIP platform as a Fault Management platform or a Network Management platform. It therefore presents either a platform model, a fault model, or a network/service model. By reaching deeper into the TeMIP model details, however, it can present a much richer model. You can then choose to include more details in the model.

4.5.1 Generating a model

The integrated platform does not include a tool for generating a service model automatically. You can design tools to extract the necessary data from the TeMIP dictionary in order to create a service model within SQM. This inevitably results in an excessively low-level service model.

Alternatively, you can base an automatic service model creation tool on the Acanthis tools used to create the database export tables, linking them only to exported data. Again, this results in an excessively low-level service model. Alternatively, once you have filled the discovery table with the entity names, you can then use a simple tool to generate an SCI xml file listing all managed entities.

The DFI discovery process provides a good basis for producing such a service model, but it is better suited to generating SCI objects that are automatically bound to the discovered DFIs. A TeMIP Service Definition is therefore needed to encompass the Service Component Definition that represents the TeMIP managed entity (these are shared globally, so that the SCIs can be reused). This method offers limited added value, but it may be of some use if you wish to generate a complete mapping model automatically, to simplify service administration.

The discovery process is based on retrieving the list of all DFIs that the SQL SA manages from a specific table. It is currently not possible to incorporate in the discovery engine a feature for generating all SCIs linked to the discovered DFI. An additional tool could use the generated DFI xml files to generate the associated SCI xml file, however.

4.5.2 Binding parameters

The integrated platform binds all parameters automatically when it generates the SCIs from each DFI it discovers. The degree of added value generated by providing the DFI’s SCIs is open to discussion. Our aim in this case is clearly to automate the binding of parameters and simplify integration at instantiation time.

4.5.3 Binding properties

Since the integrated platform generates the SCI automatically from the DFI, you can copy the DFI’s properties to each related SCI quite simply with a matching definition. By doing so, you can simplify data integration at the SLMonitoring interface and provide a drill-down feature, as there is no access to DFI data at the SLMonitoring level.

In this way, all of the parent application’s features can be used to handle data that can be accessed at the SCI level. Users can then select an SCI parameter and display all associated alarms from the underlying entity within TeMIP, using the TeMIP entity name property that is copied at instantiation time.

4.5.4 The TeMIP model within SQM

From an OSS standpoint, the TeMIP platform is a distributed application containing multiple modules interfaced with network and service element managers.

The platform model is virtually a ‘black box’ providing general parameters that do not require a detailed model. The model within SQM can be limited to a Service Definition consisting in the TeMIP platform as a whole and a Service Component Definition consisting in a TeMIP director.

In most cases, the TeMIP platform serves as a Fault Management platform that collects alarms and events on a set of entities managed by telecom operators.

The fault model consists in providing all alarms and events that the TeMIP Fault Manager collects (containment, processing and life-cycle). It concentrates on alarms that are collected and stored on managed entities in operation contexts. The alarm life-cycle, and the statistics calculated from them, is then passed to a fault collection model within SQM.

The model within SQM may consist in a Service Definition that represents the Network Operator responsible for a set of operation contexts, i.e. a set of managed entities, comprising the Service Component Definition.

The TeMIP platform can equally be seen as a Network & Service Management platform that collects and provides data on network elements and their related services. The model is then highly dependent on the model described in the TeMIP dictionary. In this case, you must define what level of detail you wish to model in SQM.

Another standard TeMIP model that can be produced is the domain hierarchy, since this consists in grouping managed entities according to various criteria such as geography, technology, responsibility, customer, etc. It may be potentially very useful to model the TeMIP domain as a Service Component Definition.

4.6 Setting up the SQL SA

There are no special considerations in setting up the TeMIP SQL SA. The only potentially specific aspects relate to loading the SQM model data, setting up the independent discovery process, and uploading the views. Each of these points is covered in the following paragraphs.

4.6.1 Uploading the default model

You must load the basic TeMIP model required by the SA into the SRM before the platform can generate any xml file. This model is only required if the discovery process also generates SCI files in XML format, to prevent model instantiation errors occurring.

4.6.2 Uploading views automatically

The *sqlexec* tool is used to push the SQL views and PL/SQL functions required in combination with the export tables. You must enter the database URL and the SQL scripts that must be loaded (in the correct order) as this tool’s arguments. The SQM acquisition director must be able to access the database.

4.6.3 Independent discovery and upload

The independent discovery process uses a table that has been created in the automatic view upload phase. Once this is done, you must execute the dedicated discovery tool on the TeMIP director to populate the discovery table in the database. Again, the SQM acquisition director must be able to access the database.

Once TeMIP discovery has been run, automatic SQL discovery can begin and generate the DFI XML files. If model instantiation is also required, you must execute or activate it after this discovery process, as it is not yet incorporated in the discovery process. You must then load all of the XML files into the Tibco repository and the SRM.

4.7 SQM tuning

Once you have loaded the model and created the instances, the SQM platform also offers tuning features enabling it to deal with special data cases such as subscriber-dependent data, Naming Service resolution and internal hooks such as auto-forwarding, naming plan flags, etc. Each of these points is covered in the following paragraphs.

4.7.1 Subscriber-dependent parameters

Some parameters may be subscriber dependent. The subscriber information may not always be available in the export data table, however.

In the case of TeMIP classes, this depends on the model concerned and whether it includes this type of information as an attribute.

In the case of TeMIP events, nothing identifies whether the event is related to a specific customer.

In the operation context attributes, the owner ID, responsible operators or responsible person can serve as subscriber information if these are filled in. In the alarm object attributes, only the domain-related parameters (if they are associated with a customer) or the operator responsible for its operation context can provide subscriber information.

4.7.2 Naming Service resolution

SQM uses the Naming Service to resolve and retrieve the customer information based on naming plans. If a parameter is subscriber dependent, the subscriber information and the subscriber domain are used to map to a customer name. Additionally, if the subscriber domain is set to *ServiceCenter*, the Data Collector does not use the Naming Service to resolve the name (and so bypasses naming resolution).

4.8 Launching user interfaces

The SLMonitoring user interface offers a launch mechanism to navigate from this interface to the user interface of other applications. It also provides a set of services for controlling SLMonitoring to activate or open a window for a service within a service level agreement.

4.8.1 Typical interfaces launched

By default, the TeMIP Gateway and the TeMIP Fault Statistics Service Adapter launch the "Display associated alarms" interface. This provides drill-up and drill-down capabilities with TeMIP Client Real Time Alarm Handling.

4.8.2 Using properties

In the SLMonitoring user interface, the user selects an SCI or SI parameter. The SLMonitoring user interface then launches the TeMIP Client user interface, focusing on the managed entity defined by the entity's name property.

This requires the properties contained within the MRP of the DFI's SCIs.

SLMonitoring cannot access information on the DFI instances and their associated data. When you are creating a model, you must take this fact into account so that all data needed to launch the TeMIP Client user interface are available in the created SCI.

Appendix A

Troubleshooting

OpenView TeMIP

For troubleshooting information on OpenView TeMIP, see the troubleshooting documentation supplied with the TeMIP product.

Acanthis KnowledgeWare

For troubleshooting information on Acanthis KnowledgeWare, see the troubleshooting documentation supplied with the Acanthis product.

OpenView SQM

For troubleshooting information on OpenView SQM, see the troubleshooting documentation supplied with the SQM product.

Appendix B

TeMIP/Oracle datatypes

OV TeMIP	ORACLE
MCC_K_DT_BIN_ABS_TIM	DATE *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_REAL	FLOAT *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_FLOATF	FLOAT *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_BITSET	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_COUNTER32	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_LCOUNTER32	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_UNSIGNED32	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_UNSIGNED32	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_COMPONENT	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_IPADDRESS	NUMBER(0..38) *
	VARCHAR2(1..4000)

OV TeMIP	ORACLE
	CHAR(1..4000)
	CLOB
MCC_K_DT_MCC_ERROR	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_OCTET	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_INTEGER8	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_UNSIGNED8	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_INTEGER16	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_UNSIGNED16	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_COUNTER16	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_LCOUNTER16	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_TIME24	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_INTEGER64	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_UNSIGNED64	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)
	CLOB
MCC_K_DT_COUNTER64	NUMBER(0..38) *
	VARCHAR2(1..4000)
	CHAR(1..4000)

OV TeMIP	ORACLE
	CLOB
Other OV TeMIP datatypes	VARCHAR2(1..4000) *
	CHAR(1..4000)
	CLOB

Glossary

The following table lists the acronyms commonly used in this document.

Term	Description
DFD	Data feeder definition
DFI	Data feeder instance
MO	Managed Object
MRP	Measurement reference point
NE	Network Element
OC	Operation Context
OV SQM	OpenView Service Quality Manager
OV TeMIP	OpenView Telecommunications Management Information Platform
SA	Similar Alarm
SAI	Service Adapter Application Name (or Service Adapter instance)
SL	Service Level
SLA	Service Level Agreement
SLM	Service Level Management