HP OpenView
Service Quality Manager

TeMIP real-time Fault Service Adapter

Installation, Configuration and User's Guide

Edition: 1.4

for the HP-UX Operating Systems

March 2007

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

Trademark Notices

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

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

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

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

UNIX® is a registered trademark of The Open Group.

Contents

Preface

This document describes how to install and use the hp OpenView Service Quality Manager (OV SQM) TeMIP Fault Service Adapter. This Service Adapter, once configured, is able to collect in real-time TeMIP Alarms impacting a Service, using a Problem management system, on top of TeMIP, in order to fit them into SQM as quality indicators.

This document describes how to:

- Install and setup a TeMIP Fault Service Adapter on the TeMIP dedicated hardware.
- Declaring the TeMIP Fault Service Adapter on the SQM platform, to allow the interconnection with TeMIP.
- Map alarms to service parameters in OV SQM.

Intended Audience

This document is intended for Service Quality Manager administrators and integrators.

Required Knowledge

It is assumed that the reader is familiar with the functionality of Service Quality Manager and has previous experience of the following:

- System administration and operations
- Service Level Management

It is assumed that the reader is familiar with the concepts described in the following books:

- *HP OpenView Service Quality Manager Overview*.
- *HP OpenView Service Quality Manager Service Adapter User's Guide.*
- *HP OpenView Service Quality Manager Administration Guide.*

Software Versions

The software versions referred to in this document are specified in chapter 2.1.

Typographical Conventions

`Courier` Font

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

Italic Text

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

Bold Text

- New terms and to emphasize important words.

Associated Documents

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

Support

You can visit the HP OpenView support web site at:

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

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

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

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

Chapter 1

Introduction

1.1 Overall solution architecture overview

HP OpenView 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. OV SQM links service quality degradations to potential effects on business, allowing network support operators to address problems and prioritize actions proactively.

OV SQM monitors the service quality by aggregating performance or quality indicators collected from various data sources, such as the network, the 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).

The TeMIP environment is extremely flexible and allows you to create custom solutions that use TeMIP in a way that suits your particular requirements.

We want to integrate OV SQM with TeMIP which consists of a suite of components that together form a Telecommunications Network Management System. These components offer a solid, modular, distributed architecture and an open development environment, and answer the need for off-the-shelf integrated network management.

The OV SQM TeMIP Fault Service Adapter is one component of the SIA (Service Impact Analysis) solution which provides a way to collect real-time TeMIP Alarms impacting a Service handled by OV SQM.

This document first describes the way the OV SQM TeMIP Fault Service Adapter should be installed and configured on the TeMIP dedicated director.

Then it explains how the TeMIP Fault Service Adapter should be configured on the OV SQM platform, to enable the connection.

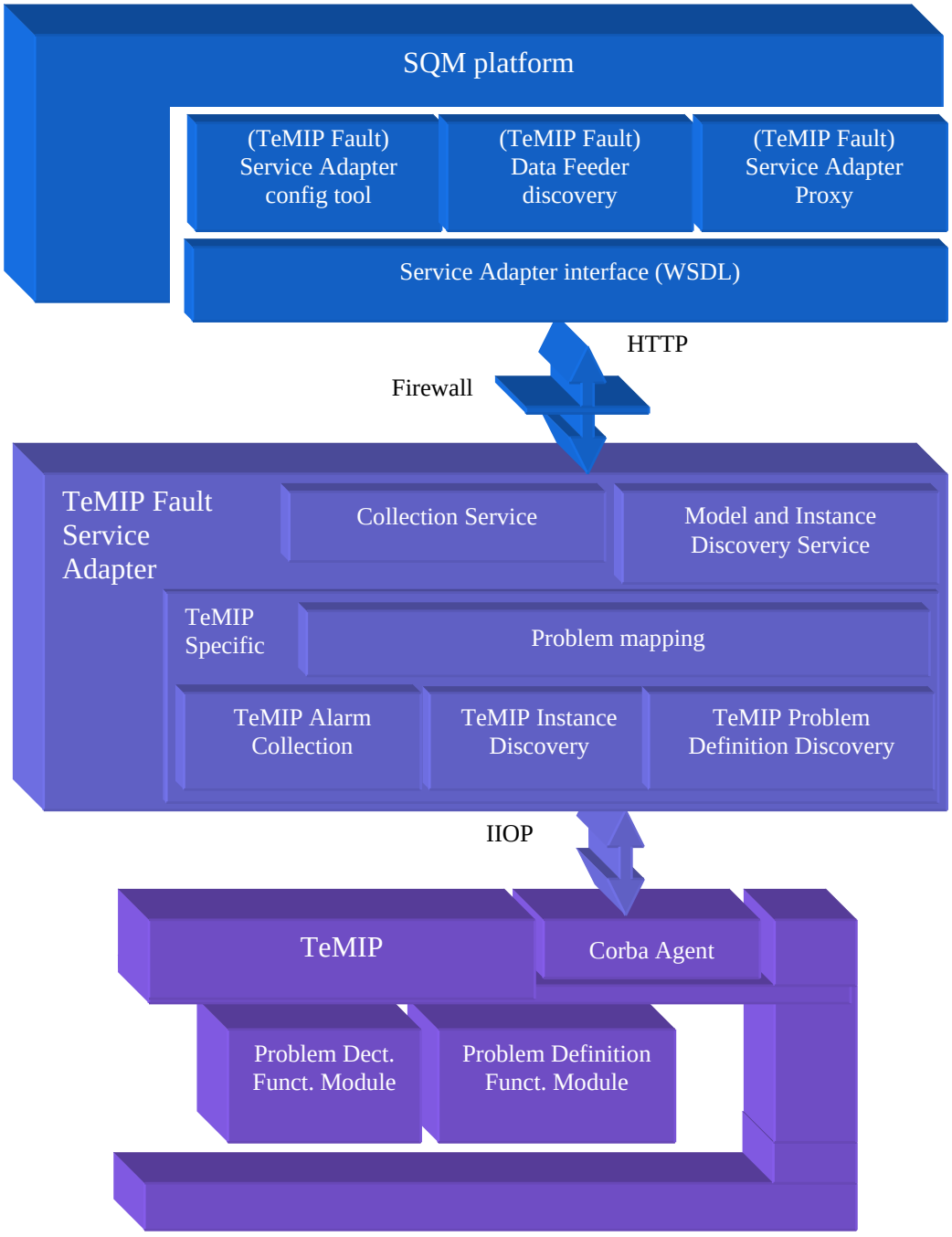
Finally, it shows how to enable alarms to impact services in OV SQM, by providing a description of the mapping/association of TeMIP Alarms against OV SQM Data Feeders.

1.1.1 Software architecture

The TeMIP Fault Service Adapter is a middle-tier, which adapts a Problem management system, built on top of TeMIP, to make it fit into OV SQM as a quality indicator data source. The main role is to interface with the OV SQM southbound interface, using all available features provided by the TeMIP Problem Detection Functional Module, accessed through the TeMIP CORBA Agent.

This integration is illustrated in the following figure, “TeMIP/SQM integration”.

Figure 1 TeMIP/SQM Integration



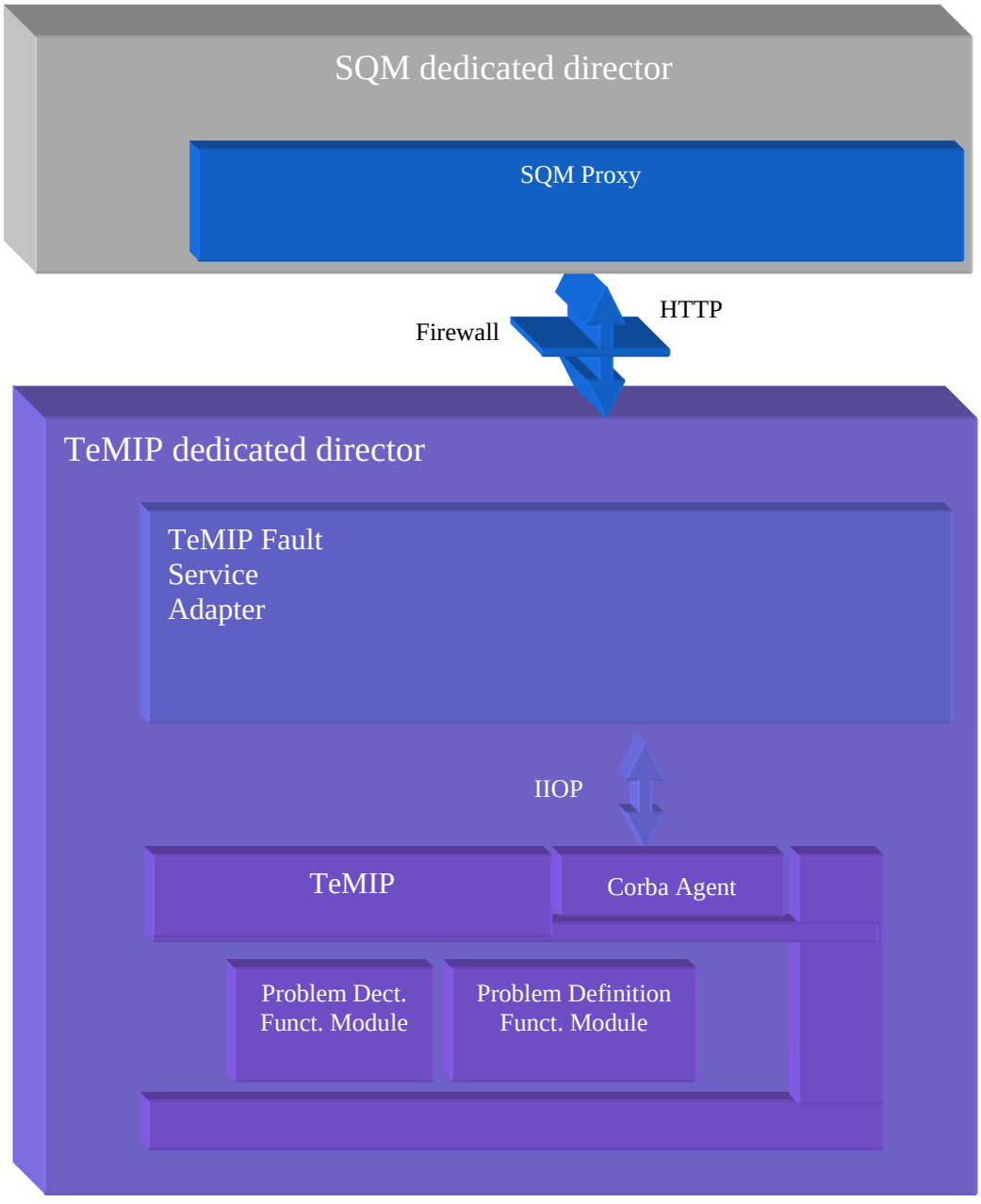
1.2 Director architecture

The following diagram describes the recommended deployment of the overall solution. The best practice is to install and configure the TeMIP Fault Service Adapter on the same hardware as the existing TeMIP platform. Indeed, the minimal requirement is that the

TeMIP platform shares the file system with the TeMIP Fault Service Adapter, as some resources, like the CORBA IOR (Interoperable Object Reference) file, are vital for the solution.

This integration is illustrated in the following figure, “TeMIP/SQM director architecture”.

Figure 2 TeMIP/SQM Director Architecture



Chapter 2

Installation

2.1 Software prerequisites

As explained in the former chapter, the TeMIP Fault Service Adapter is a "middle-tier", whose role is to use a Problem management system, implemented on top of TeMIP, as a quality indicator data source for OV SQM.

Therefore, the TeMIP Fault Service Adapter requires the following operational platforms:

- OV SQM V1.4 (or more recent)
- TeMIP V5.2 (or more recent)

The TeMIP Fault Service Adapter also requires some third party products, which have to be installed and setup. In case these products have already been installed, they need to be setup according to the recommendations. we could refer to chapter 2.1.3 for more information.

2.1.1 Installing Open View SQM (SQM dedicated director)

Please refer to the OV SQM Installation Guide for further details.

Note

Double check that the following kit has been also been installed:

- Service Adapter Proxy

Note that this kit is not installed by default. Refer to the setup chapter for configuration details

2.1.2 Installing Open View TeMIP (TeMIP dedicated director)

Please refer to the OV TeMIP Installation Guide for further details.

Note

Double check that the following kit has been also been installed:

- TeMIP CORBA Agent V5.0

Note that this kit is not installed by default.

2.1.3 Installing Pre-requisite Software (TeMIP dedicated director)

Note

Remember that the Third Party Products, required for the TeMIP Fault Service Adapter, have to be installed on the TeMIP dedicated director.

Note that these third party products, listed below, have to be installed on the director, which will run the TeMIP Fault Service Adapter application itself.

Thus, in addition to the OV SQM and TeMIP operational platforms, this Service Adapter also requires the following products, which should be installed in the same order as they appear in the table:

Product	Version	**Description**
HPOvLcore.HPOVXPL	2.61.110	**HP OpenView Cross Platform Component**
HPOvJext.HPOVJXPL	2.61.110	**HP OpenView Cross Platform Component Java**
HPOvLcore.HPOVSECCO	2.00.110	**HP OpenView Security Core**
HPOvLcore.HPOVBBC	5.00.121	**HP OpenView HTTP Communication**
HPOvLcore.HPOVCTRL	1.50.111	**HP OpenView Process Control**
HPOvAcc.HPOVJDKA	1.04.200	**JDK version 1.4.2_02 for HP OpenView**
HPOvAcc.HPOVTOMCATA	5.00.284	**HP OpenView Tomcat Servlet Container -V5.0.28- Package Version 05.00.284**

Note that this includes the following versions:

- Java 2 Standard Edition Software Development Kit (SDK). Recommended release: 1.4.2_13
- Apache Jakarta Tomcat (Web application container). Recommended release: 5.00.284-

To install the pre-requisite packages, use the standard swinstall tool.

This command installs all filesets within a depot:

```
swinstall -s /path/to/file.depot \*
```

```
Example: # swinstall –s /tmp/HPOvXpl-02.61.110-HPUX11.0-
release.depot \*
```

2.2 Installing the TeMIP Fault Service Adapter (TeMIP dedicated director)

Note

Remember that the TeMIP Fault Service Adapter has to be installed on the TeMIP dedicated director.

To install the TeMIP Fault Service Adapter by performing the following steps:

- First, log on to the system as **root** user.
- Mount the HP OpenView Service Adapters and Gateways CD-ROM on your system.
- Go to <*mount directory*>/SQM-1.40.00
- Set the TEMIP_SC_HOME environment variable to the SQM Root directory:

```
# export TEMIP_SC_HOME=<SQM installation directory>
```

- Install the TeMIP Fault Service Adapter InstallAnywhere kit.

```
# SQMSATEMIPFAULT-1.40.00.bin
```

This package delivers the following data tree:

```
TFSA/v1_0/bin/temip_fault_modify_ao.sh
TFSA/v1_0/bin/temip_fault_sa_configure.sh
TFSA/v1_0/bin/temip_fault_dict_extractor.sh
TFSA/v1_0/bin/temip_sc_dict_extractor
TFSA/v1_0/conf/TeMIPDictionary.properties
TFSA/v1_0/conf/TeMIPFaultSA_Runtime.properties
TFSA/v1_0/war/TeMIPFaultSA.war
```

A short description of the directories of the release:

- TFSA/v1_0: “ v1_0” stands for the TeMIP Fault Service Adapter version
- TFSA/v1_0/war: contains Service Adapter war file, which has to be deployed on the Web application container (Tomcat), using the configuration script.
- TFSA/v1_0/bin: (scripts)
  - the configuration script
  - the TeMIP class instance dictionary extraction
  - the TeMIP alarm_object extension script
- TFSA/v1_0/conf: (configuration data)
  - the default TFSA configuration files

Chapter 3

Setting up and Configuration

The configuration requires that all the previous installation steps have been done.

The TeMIP Fault Service Adapter configuration is decomposed in three phases. Each phase being performed on their associated director. These phases are:

1. Configuration of the TeMIP Fault Service Adapter.
2. Deploy the TeMIP Fault Service Adapter within the Tomcat web application container, which provides the execution environment
3. Setup the OV SQM Service Adapter Proxy and provide the information required for the connection with the TeMIP Fault Service Adapter.

3.1 Configuring the TeMIP Fault Service Adapter (TeMIP dedicated director)

Note

The 1.0 release of the TeMIP Fault Service Adapter requires that the variable data is stored and accessible by the root user, through the fix link: /var/opt/temip

3.1.1 Configuring

Use the temip_fault_sa_configure.sh script to generate the configuration data file.

Command:

- Connect as “root” user.
- Perform the following commands to create the TeMIP Fault SA config file

```
# cd /opt/OV/TFSA/v1_0/bin
# ./temip_fault_sa_configure.sh
```

Output (including interactive prompts):

```
Starting TeMIP Fault SA configuration...
Configuration succeeded.
```

Check that Tomcat port is available:

```
# ovtomcatctl -getconf
EnableHTTP=False
EnableHTTPS=False
EnableJk2Ajp13=True
HTTPPort=8080
HTTPSPort=8443
Jk2Ajp13Port=8009
ShutdownPort=8005
# ovtomcatctl -checkport 8080
```

If it is not available, either make it so (by moving the application that is using it), or use another port with Tomcat. If another port is used, the SA Proxy will need to be configured appropriately as well.

Tomcat logs appear in /opt/OV/nonOV/tomcat/a/logs. If the TeMIP Fault SA is correctly configured the message " TeMIP Fault SA Starting up" should be seen in the tomcat log.

3.1.2 Providing TeMIP class instances dictionary entries

Note

In case the following message " DISCARD notification on UNKNOWN MANAGED CLASS!" appears within the log file, this extraction operation has to be performed again.

Note that in case the OV SQM Service Adapter Proxy is already running, it has to be stopped and started again in order to force the TeMIP Fault SA to take into account the newly extracted data.

Use the temip_fault_sa_temip_dict_extract.sh script to extract the class definitions from the TeMIP dictionary for which a managed network resource problem could be reported.

Command:

- Connect as "root" user.
- You can use the dictionary browser to find the class IDs.
- Perform the following commands to create the TeMIP Fault SA config file

```
# cd /opt/OV/TFSA/1.0/bin
# ./temip_fault_sa_temip_dict_extract.sh <class Id> { <class
Id> }
```

Example:

To allow the TeMIP Fault Service Adapter to handle for example the OSI_SYSTEM managed entity and child entities, perform the following command:

```
# cd /opt/OV/TFSA/1.0/bin
# ./temip_fault_sa_temip_dict_extract.sh 26
```

3.1.3 Adding TeMIP alarm_object extended attributes

Note

In case the following attributes are already present for the alarm_object in the dictionary browser, do **NOT** proceed with this step. This can be checked by using the mcc_dap_browser, and in the browser window expanding the OPERATION_CONTEXT and then alarm_object.

Id	Name
10003	Source Entity
10004	Problem Category

Because the TeMIP Fault Service Adapter requires some extra alarm attributes for its operation these must be added. Use the temip_fault_modify_ao.sh script to extend the alarm_object class with the additional attributes.

Command:

- Connect as "root" user.
- Perform the following commands to create the TeMIP Fault SA config file

```
# cd /opt/OV/TFSA/1.0/bin
# ./temip_fault_modify_ao.sh
```

Example:

```
# cd /opt/OV/TFSA/1.0/bin
# ./temip_fault_modify_ao.sh

Before you run this script please make sure that the
attributes are not present in the TeMIP dictionary.

Id     Name
10003  Source Entity
10004  Problem Category

Enter Y to continue.
Y

#
```

3.2 Setting up the OV SQM Service Adapter proxy (SQM dedicated director)

This chapter does not describe these steps in detail. For complete information on how to configure the SA proxy application see the SQM Fault SA Proxy User' s Guide. Follow the steps to create an SA proxy application instance.

Simply put, the SA proxy needs to be configured to use the Tomcat port to access the TeMIP Fault SA. This is accomplished by creating a " connector" .

3.2.1.1 Associating SA Proxy to Service Adapters through connectors

The configuration consists in creating or deleting connectors (URLs) that associate the SA proxy application to existing Service Adapters (Web Services).

A connector thus contains the parameters (or URL) that allows the localization (access) to a Service Adapter. The connector is identified by a unique name, and its configuration is loaded into the OV SQM Central Repository.

Associate a new Service Adapter (create a connector)

A good naming convention for the connector is <hostname>_TFSA_<SaVersion>.

Command:

- Connect as “sqmadm” user.
- Load the OV SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)
- Perform the following command:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin
# temip_sc_configure.sh –applicationName <application name> –
dirName <director name> -addConnector <Connector name>
-----------------------------------------------------------------
----
   where:
    the <connector name> of the connector that designates a
Service Adapter.
    the <application name> is the one that has been provided at
the application setup.
    the <director name> is the director on which has been created
the application at the setup phase. (by default the director name
is acquisition).
```

This command will prompt the user for the Service Adapter (Web Service) location parameters: SA host name (including the domain name), Web Service port number and SA name (for example TeMIPFaultSA). The command not only loads these parameters into the OV SQM Central Repository, but also uses the connector name to extend the application data tree as follows:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<Application
name>/<Connector name>
```

Output (including interactive prompts):

```
Add connector "<connector name>" to "<application name>"
application ...

Create the Connector datatree.

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/SAConfig   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/filter   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector
name>/discovery/filter/slmv14_acquisition_SAProxy_OL_filter.sh
(created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/inventory   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/inventory/raw   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/inventory/filtered   (created)

/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/repository   (created)

Warning: parameter config file
/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/config/<connector name>.cfg not found

Please enter the Service Adapter (Web Service) Name:TeMIPFaultSA

Please enter the Service Adapter (Web Service) Host Name:
habine.vbe.cpqcorp.net

Please enter the Service Adapter (Web Service) Port Number: 8080
Load the Connector in the Tibco Repository
INFO: Backup written at the following location:
/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/repository.2005_4_28_17_58_51
INFO:
/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/repository/connectors_data.exp has been imported into the
Repository
INFO: Backup written at the following location:
/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/repository.2005_4_28_17_59_08
INFO:
/var/opt/OV/SQM/slmv14/ServiceAdapters/Proxy/v1_4/<application
name>/repository/monitored_connectors_data.exp has been imported
into the Repository
Add Connector succeed.
```

For listing and removing connectors, see the SA proxy User's Guide.

3.2.1.2 Checking connector availability

Command:

- Connect as "**sqmadm**" user.
- Load the OV SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)
- Perform the following command:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin
# temip_sc_discovery.sh -application <application name> -director
<director name> -platform <platform name> -connector <connector
name> -check
-------------------------------------------------------------------
-----
   where:
    the <connector name> of the connector that designates the
Service Adapter on which the Data Feeder Definitions have to be
discovered. This connector has been declared during the SA proxy
application configuration.
    the <platform name> is the one that has been defined at the
SQM Server setup and available in the variable ($KERNEL_ID).
    the <director name> is the director on which has been created
the application at the setup phase. (by default the director name
is acquisition).
    the <application name> is the one that has been provided at
the application setup.
```

This command checks the specified connector availability by sending requests to the remote Service Adaptor. It allows:

- To validate that the connector parameters (hostname, service adapter name and port number) are well set
- To validate that the remote Web Container is running
- To validate that the remote Service Adapter is deployed

3.2.2 Discovering and loading Data Feeder Definitions (DFDs)

The DFD discovery is an important feature provided by the Service Adapter proxy. The discovery indeed retrieves the DFD exposed by the Service Adapters that has been associated to this SA proxy application, during the connector creation. These DFD are then automatically loaded in the OV SQM Service Repository Manager.

Discovery script

The discovery script is located in the following directory:

`$TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin/temip_sc_discovery.sh`

Script Usage

temip_sc_discovery.sh -connector <value> -platform <value> -director <value> -application <value> -dfd (-discover | -load | -all)

The discovery parameters:

- -connector: The name (id) of the connector that designates the Service Adapter on which the Data Feeder Definitions have to be discovered. This connector has been declared during the SA proxy application configuration.
- –application: the SA Proxy application name defining the provided connector
- –platform: the platform's name the application belong to
- -directory: the director's name the application belong to
- –discover: perform the discovery phase only
- –load: performs the loading phase only: discovered DFDs are loaded in the SRM
- –all: perform discovery and loading phases

Script Options

The script supports the following options:

- -repoUrl: This option set the repository location, even if already defined by TEMIP_SC_REPOSITORY_LOCATION system environment variable.
- -configUrl: This option set the repository configuration url. If this option is not used, default value is /tibco/private/adapter/ServiceCenter/ServiceAdapters/

The discovery is done in 2 steps for a DFD:

- Raw discovery phase: retrieves all the DFDs which have been discovered on the Service Adapter designated through the connector name.
- Loading phase, that will load the discovered DFDs into OV SQM repository

Note

The next chapters will describe in details each phase presented above.

The same processing can be done in a single command (with a default loading of all discovered Data Feeder Definitions). Please refer to chapter 3.2.2.3 **One shot discovery and loading** for more details on this command.

3.2.2.1 Raw discovery phase

This initial phase will retrieve the DFD exposed by the Service Adapters that has been associated to this SA proxy application, during the connector creation.

Command

The discovery request has to be performed as follows:

- Connect as "sqmadm" user.
- Load the OV SQM environment variables

  (default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)

- Perform the following commands:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin

# temip_sc_discovery.sh -platform <platform name> -director
<director name> -application <application name> -connector
<connector name> -dfd -discover

-----------------------------------------------------------------
-----

   where:

    the <connector name> of the connector that designates the
Service Adapter on which the Data Feeder Definitions have to be
discovered. This connector has been declared during the SA proxy
application configuration.

    the <platform name> is the one that has been defined at the
SQM Server setup and available in the variable ($KERNEL_ID).

    the <director name> is the director on which has been created
the application at the setup phase. (by default the director name
is acquisition).

    the <application name> is the one that has been provided at
the application setup.
```

Output

The raw discovery phase output will generate the following files.

- The discovered DFD xml files that could be used to manually add or remove the DFD into the SRM, located in:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4**<application name>/<connectorname>**/discovery/repository/**DelDFDReq_<DFDName>.<DFDVersion>**.xml

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4**<application name>/<connectorname>**/discovery/repository/**NewDFDReq_<DFDName>.<DFDVersion>**.xml

NewDFDReq_TeMIP_Fault_DFD.v1_0.xml example:

```
<?xml version="1.0" encoding="UTF-8"?>
<sc:NewDFDReq xmlns:sc="http://www.compaq.com/TeMIP/ServiceCenter"
msg.id="14">
  <sc:DataFeederDef dfd.name="TeMIP_Fault_DFD" dfd.label="TeMIP Fault DFD"
dfd.version="v1_0" sa.name="Proxy" sa.label="Proxy" sa.version="v1_4">
    <sc:Descr>TeMIP Fault DFD for service impact.</sc:Descr>
    <sc:MRPNamingSchema>
      <sc:PropertyName property.name="NR" property.label="Network
Resource" />
      <sc:PropertyName property.name="OC" property.label="Operation
Context" />
    </sc:MRPNamingSchema>
    <sc:PropertyDefs>
      <sc:PropertyDef property.name="NR" property.label="Network Resource"
datatype="String">
        <sc:Descr>The network resource</sc:Descr>
      </sc:PropertyDef>
      <sc:PropertyDef property.name="OC" property.label="Operation
Context" datatype="String">
        <sc:Descr>The operation context</sc:Descr>
      </sc:PropertyDef>
      <sc:PropertyDef property.name="WSConnector"
property.label="WSConnector" datatype="String">
        <sc:Descr>FOR SA Proxy INTERNAL USE, DO NOT REMOVE OR MODIFY THIS
PROPERTY</sc:Descr>
      </sc:PropertyDef>
    </sc:PropertyDefs>
    <sc:ParameterDefs>
      <sc:ParameterDef parameter.name="Communicat_Sever"
parameter.label="CommunicationsAlarm Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>CommunicationsAlarm Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Environmen_Sever"
parameter.label="EnvironmentalAlarm Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>EnvironmentalAlarm Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="EquipmentA_Sever"
parameter.label="EquipmentAlarm Severity" datatype="Int" category="Other"
partition="QoS" customerDepend.flag="False">
        <sc:Descr>EquipmentAlarm Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="IntegrityV_Sever"
parameter.label="IntegrityViolation Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>IntegrityViolation Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Operationa_Sever"
parameter.label="OperationalViolation Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>OperationalViolation Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="PhysicalVi_Sever"
parameter.label="PhysicalViolation Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
```

```
        <sc:Descr>PhysicalViolation Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Processing_Sever"
parameter.label="ProcessingErrorAlarm Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>ProcessingErrorAlarm Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="QualityofS_Sever"
parameter.label="QualityofServiceAlarm Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>QualityofServiceAlarm Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="SecuritySe_Sever"
parameter.label="SecurityServiceOrMechanismViolation Severity"
datatype="Int" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>SecurityServiceOrMechanismViolation Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="TimeDomain_Sever"
parameter.label="TimeDomainViolation Severity" datatype="Int"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>TimeDomainViolation Severity</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Communicat_Count"
parameter.label="CommunicationsAlarm Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>CommunicationsAlarm Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Environmen_Count"
parameter.label="EnvironmentalAlarm Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>EnvironmentalAlarm Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="EquipmentA_Count"
parameter.label="EquipmentAlarm Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>EquipmentAlarm Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="IntegrityV_Count"
parameter.label="IntegrityViolation Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>IntegrityViolation Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Operationa_Count"
parameter.label="OperationalViolation Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>OperationalViolation Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="PhysicalVi_Count"
parameter.label="PhysicalViolation Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>PhysicalViolation Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Processing_Count"
parameter.label="ProcessingErrorAlarm Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>ProcessingErrorAlarm Occurrence</sc:Descr>
      </sc:ParameterDef>
```

```
      <sc:ParameterDef parameter.name="QualityofS_Count"
parameter.label="QualityofServiceAlarm Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>QualityofServiceAlarm Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="SecuritySe_Count"
parameter.label="SecurityServiceOrMechanismViolation Occurrence"
datatype="Int" category="Counter" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>SecurityServiceOrMechanismViolation
Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="TimeDomain_Count"
parameter.label="TimeDomainViolation Occurrence" datatype="Int"
category="Counter" partition="QoS" customerDepend.flag="False">
        <sc:Descr>TimeDomainViolation Occurrence</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Communicat_Info"
parameter.label="CommunicationsAlarm Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>CommunicationsAlarm Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Environmen_Info"
parameter.label="EnvironmentalAlarm Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>EnvironmentalAlarm Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="EquipmentA_Info"
parameter.label="EquipmentAlarm Incident Information" datatype="String"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>EquipmentAlarm Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="IntegrityV_Info"
parameter.label="IntegrityViolation Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>IntegrityViolation Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Operationa_Info"
parameter.label="OperationalViolation Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>OperationalViolation Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="PhysicalVi_Info"
parameter.label="PhysicalViolation Incident Information" datatype="String"
category="Other" partition="QoS" customerDepend.flag="False">
        <sc:Descr>PhysicalViolation Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="Processing_Info"
parameter.label="ProcessingErrorAlarm Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>ProcessingErrorAlarm Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="QualityofS_Info"
```

```
parameter.label="QualityofServiceAlarm Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>QualityofServiceAlarm Incident Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="SecuritySe_Info"
parameter.label="SecurityServiceOrMechanismViolation Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>SecurityServiceOrMechanismViolation Incident
Information</sc:Descr>
      </sc:ParameterDef>
      <sc:ParameterDef parameter.name="TimeDomain_Info"
parameter.label="TimeDomainViolation Incident Information"
datatype="String" category="Other" partition="QoS"
customerDepend.flag="False">
        <sc:Descr>TimeDomainViolation Incident Information</sc:Descr>
      </sc:ParameterDef>
    </sc:ParameterDefs>
  </sc:DataFeederDef>
</sc:NewDFDReq>
```

After this discovery phase, it is possible to update the Data Feeder Definition by updating the XML files.

Note

Each time that a Data Feeder Discovery is performed, the XML files located in the directory are backed-up to avoid loading old DFD XML files during the loading phase.

3.2.2.2 Loading phase

Command

The discovery loading request has to be performed as follows:

- Connect as “sqmadm” user.
- Load the OV SQM environment variables

(default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin

# temip_sc_discovery.sh -platform <platform name> -director
<director name> -application <application name> -connector
<connector name> -dfd -load

-------------------------------------------------------------------
-----

   where:

    the <connector name> of the connector that designates the
Service Adapter for which the Data Feeder Definitions or Instances
have to be discovered. This connector has been declared during the
SA proxy application configuration.

    the <platform name> is the one that has been defined at the
```

```
SQM Server setup and available in the variable ($KERNEL_ID).

    the <director name> is the director on which has been created
the application at the setup phase. (by default the director name
is acquisition).

    the <application name> is the one that has been provided at
the application setup.
```

The command loads into the OV SQM Service Repository Manager the Data Feeder Definition (XML files) located in:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4<application name>/<connectorname>/discovery/repository/`

3.2.2.3 One shot discovery and loading

If the user does not want to call separately the DFD discovery steps described above (discovery and load), the DFD discovery can be performed in a single command, as described below:

Command

- Connect as “sqmadm” user.
- Load the OV SQM environment variables

  (default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)
- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin

# temip_sc_discovery.sh -platform <platform name> -director
<director name> -application <application name> -connector
<connector name> -dfd -all


-----------------------------------------------------------------
-----

   where:

    the <connector name> of the connector that designates the
Service Adapter for which the Data Feeder Definitions or Instances
have to be discovered. This connector has been declared during the
SA proxy application configuration.

    the <platform name> is the one that has been defined at the
SQM Server setup and available in the variable ($KERNEL_ID).

    the <director name> is the director on which has been created
the application at the setup phase. (by default the director name
is acquisition).

    the <application name> is the one that has been provided at
the application setup.
```

Output

The discovery will perform:

- The raw DFD discovery request
- Load all the discovered DFDs into the OV SQM Service Repository Manager

3.2.3 Discovering and loading Data Feeder Instances (DFIs)

The DFI discovery is an important feature provided by the Service Adapter proxy. The discovery retrieves the DFIs exposed by the Service Adapters that has been associated to this SA proxy application, during the creation of connector. These DFIs are then be automatically loaded into the SQM Service Repository Manager.

Note

Before DFI Discovery is performed, you must fill data for one item: operation_context in below property file:
/var/opt/OV/TFSA/v1_0/properties/TeMIPFaultSA_Runtime.properties

temip.operation_context=<put Temip Operation Context into here and spreate it by "." eg ".xlm,.octest">

Discovery script

The discovery script is located in the following directory:

```
$TEMIP_SC_HOME/ServiceAdapters/SaProxy/v1_4/bin/temip_sc_discovery
.sh
```

Script Usage

```
temip_sc_discovery.sh –dfi -connector <connector name> -platform
<platform name> -director <director name> -application
<application name> (-discover | -filter | -load [-diff (no|
reffile | srm)] | -all )
```

The discovery parameters:

- -connector: The name (id) of the connector that designates the Service Adapter on which the Data Feeder Definitions have to be discovered. This connector has been declared during the SA proxy application configuration.
- –application: the SA Proxy application name defining the provided connector
- –platform: the platform's name the application belong to
- -director： the director's name the application belong to
- –discover: performs the discovery phase only
- -filter: performs the discovery filtering phase only.
- –load: performs the loading phase only, Discovered DFIs are loaded in the SRM
- -diff: allows specifying the options of the loading phase (default: -diff no)
- –all: perform discovery, filtering and loading phases

The discovery is done in 3 steps for DFIs:

- Raw discovery phase: that retrieves all the DFIs which have been discovered on the Service Adapter designated through the connector name, into a raw inventory file.
- Filtering phase: that executes a user-defined script that will filter the DFIs declared in the raw inventory file. It will generate a new filtered inventory file with only the desired DFIs to be managed by the application.
- Loading phase: that will load the filtered DFIs into SQM repository, base on 3 algorithms:
  - -diff no

    This option will load all the filtered Data Feeder Instances into SQM repository.
  - -diff offline

    This option will compare the list of discovered/filtered Data Feeder Instances to a discovery reference file (provided by the user).

    If a Data Feeder Instance exists in the inventory file but does not exist in the reference file, the Data Feeder instance is created.

    If the Data Feeder Instance does not exist in the inventory file but exists in the reference file, the Data Feeder is deleted from the SQM repository.

    If the Data Feeder Instance exists in both the inventory file and the reference file, it will not be reloaded.
  - -diff online

    This option performs the same Data Feeder Instances comparisons as the ***offline*** mode, but instead of considering a reference file, the declaration will depend on the existence of the Data Feeder Instance in SQM.

Note

The next chapters will describe in details each phase presented above.

The same processing can be done in a single command (with a default loading of all filtered Data Feeder Instances: **-diff no**). Please refer to chapter 3.2.3.4 **One shot discovery and loading** for more details on this command.

3.2.3.1 Raw discovery phase

This initial phase will retrieve the DFIs exposed by the Service Adapters that has been associated to this SA proxy application, during the connector creation.

Command

- The discovery request has to be performed as follows:
- Connect as "**sqmadm**" user.
- Load the SQM environment variables

  (default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)

- Perform the following commands:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin

# temip_sc_discovery.sh –platform <platform name> –director
<director name> -application <application name> -connector
<connector name>-dfi -discover

-----------------------------------------------------------------
-------

where:

the <connector name> of the connector that designates the
Service Adapter on which the Data Feeder Definitions or
Instances have to be discovered. This connector has been
declared during the SA proxy application configuration.

the <platform name> is the one that has been defined at the SQM
Server setup and available in the variable ($KERNEL_ID).

the <director name> is the director on which has been created
the application at the setup phase. (by default the director
name is acquisition).

the <application name> is the one that has been provided at the
application setup.
```

Output

The raw discovery phase output will generate the following files.

- The discovered DFI inventory file, located in:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/inventory/raw/<platform
name>_<director name>_<application name>.xml
```

- The associated DFI XML files that could be used to manually add or remove the DFI into the SRM, located in:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application
name>/<connectorname>/discovery/repository/DeclareDFIReq_<DFDName>
.<DFDversion>.<DFIID>.xml

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4<application
name>/<connectorname>/discovery/repository/DelDFIReq_<DFDName>.<DF
Dversion>.<DFIID>.xml
```

3.2.3.2 Filtering phase

The discovery filtering phase consists in creating a filtering script that will be launched by the discovery tool.

This filtering script will parse the raw discovery file (output of the previous command). The filtering script will remove the DFI definitions that will not be managed by the SQL SA application.

This filtering is mainly used for load balancing (share the DFI load on several SQL SA applications).

This script will generate a new DFI inventory file containing only the DFIs that the SQL SA application will manage.

By default, a filtering script is provided with the SQL SA Customization, and this script only copy the input raw inventory file to the filtered inventory file, without any processing.

The user/integrator will have to customize this script if necessary.

Input

The filtering script is located at:

On Unix:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAversion>/discovery/filter/<platform name>_<director name>_<application name>_filter.sh`

On Windows:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\discovery\filter\<platform name>_<director name>_<application name>_filter.bat`

Note

The filtering script can be customized by the integrator. The script accepts two input arguments:

- Raw inventory file name (full path of the raw inventory file)
- Filtered inventory file name (full path of the file that will be generated by the script).

An example of filtering script is provided in Appendix .

The raw DFI inventory file is located at:

On Unix:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAVersion>/discovery/inventory/raw/<platform name>_<director name>_<application name>.xml`

On Windows:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAVersion>\discovery\inventory\raw\<platform name>_<director name>_<application name>.xml`

Command

The discovery filtering request has to be performed as follows:

On Unix:

- Connect as “sqmadm” user.
- Load the SQM environment variables

  (default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)
- Perform the following commands

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAversion>
/bin

#temip_sc_discovery.sh –platform <platform name> –director
<director name> -application <application name> -filter


-------------------------------------------------------------------
-----

     where:

        the <platform name> is the one that has been defined at
the SQM Server setup and available in the variable ($KERNEL_ID).

        the <director name> is the director on which has been
created the application at the setup phase. (by default the
director name is acquisition).

        the <application name> is the one that has been provided
at the application setup.
```

On Windows:

Open a Command line window:

```
# cd
“%TEMIP_SC_HOME%”\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversi
on>\bin

#temip_sc_discovery –platform <platform name> -director <director
name> -application <application name> -filter


-------------------------------------------------------------------
------

     where:

        the <platform name> is the one that has been defined at
the SQM Server setup and available in the variable (%KERNEL_ID%).

        the <director name> is the director on which has been
created the application at the setup phase. (by default the
director name is acquisition).
```

```
        the <application name> is the one that has been provided
at the application setup.
```

Output

Once the raw DFI discovery file is filtered, the script will generate the filtered inventory file into:

On Unix:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/<SACustomName>_<SAversion>/discovery/inventory/filtered/<platform name>_<director name>_<application name>.xml`

On Windows:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\discovery\inventory\filtered\<platform name>_<director name>_<application name>.xml`

3.2.3.3 Loading phase

Depending on the **"-diff**" option provided when launching the discovery script, the following actions will be performed (by default the option "-diff no" is used to load all filtered Data Feeder Instances):

- -diff no

This option will load all the filtered Data Feeder Instances into SQM repository.

- -diff offline

This option will compare the list of discovered/filtered Data Feeder Instances against a DFI reference file. The reference file must be located in:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAversion>/discovery/repository/<platform name>_<director name>_<application name>_discovery_reference.xml`

This reference file must be managed manually by the user.

If a Data Feeder instance exists in the inventory file but does not exist in the reference file, the Data Feeder instance is created.

If the Data Feeder Instance does not exist in the inventory file but exists in the reference file, the Data Feeder is deleted from the SQM repository.

If the Data Feeder Instance exists in both (inventory file and reference file), it will not be reloaded.

- -diff online

This option performs the same Data Feeder Instances comparisons as the ***offline*** mode, but instead of considering a reference file, the declaration will depend on the existence of the Data Feeder Instance in SQM.

Input

The DFI filtered inventory file (output from the previous command) is mandatory as input for this phase.

It is available at:

On Unix:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/<SACustomName>_<SAversion>/discovery/inventory/filtered/<platform name>_<director name>_<application name>.xml`

On Windows:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\discovery\inventory\filtered\<platform name>_<director name>_<application name>.xml`

The inventory reference file may be necessary for the loading option**: -diff offline**.

The file must be present at the following location:

On Unix:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAversion>/discovery/repository/<platform name>_<director name>_<application name>_discovery_reference.xml`

On Windows:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\discovery\repository\<platform name>_<director name>_<application name>_discovery_reference.xml`

Command

The discovery loading request has to be performed as follows:

On Unix:

- Connect as “sqmadm” user.
- Load the SQM environment variables

(default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)

- Perform the following commands

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_4/<SACustomName>_<SAversion>/bin

#temip_sc_discovery.sh -platform <platform name> -director
<director name> -application <application name> -load -diff [no
| offline | online]


-----------------------------------------------------------------
-------

     where:
```

the ***<platform name>*** is the one that has been defined at the SQM Server setup and available in the variable ($KERNEL_ID).

the ***<director name>*** is the director on which has been created the application at the setup phase. (by default the director name is **acquisition**).

the ***<application name>*** is the one that has been provided at the application setup.

On Windows:

Open a Command line window:

```
# cd
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\bin

#temip_sc_discovery -platform <platform name> -director
<director name> -application <application name> -load -diff [no
| offline | online]


-----------------------------------------------------------------
--------

     where:

        the <platform name> is the one that has been defined at
the SQM Server setup and available in the variable
(%KERNEL_ID%).

        the <director name> is the director on which has been
created the application at the setup phase. (by default the
director name is acquisition).

        the <application name> is the one that has been provided
at the application setup.
```

Output

The status of each DFI loading (Successful, Failure, partial) will be logged.

The discovery loading procedure will log the result of each DFI declaration into:

On Unix:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/***<SACustomName>_<SAversion>***/discovery/repository***/<platform name>_<director name>_<application name>***_discovery_cmds.log

On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4***<SACustomName>_<SAversion>***\discovery\repository***<platform name>_<director name>_<application name>***_discovery_cmds.log

In case of failure, the following script can be run manually by the user, to restart the DFI loading process:

On Unix:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/***<SACustomName>_<SAversion>***/discovery/repository/***<platform name>_<director name>_<application name>***_discovery_cmds.sh

On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_4\<SACustomName>_<SAversion>\discovery\repository***<platform name>_<director name>_<application name>***_discovery_cmds.bat

3.2.3.4 One shot discovery and loading

If the user does not want to call separately the DFI discovery steps described above (discover, filter, load), the DFI discovery can be performed in a single command, as described below:

Command

- Connect as “sqmadm” user.
- Load the SQM environment variables

(default: /var/opt/OV/SQM/slmv14/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_4/bin

#temip_sc_discovery.sh –platform <platform name> –director
<director name> -application <application name> -connector
<connector name> -dfi -all

-----------------------------------------------------------------
-------

where:

the <connector name> of the connector that designates the
Service Adapter for which the Data Feeder Instances have to be
discovered. This connector has been declared during the SA proxy
application configuration.

the <platform name> is the one that has been defined at the SQM
Server setup and available in the variable ($KERNEL_ID).

the <director name> is the director on which has been created
the application at the setup phase. (by default the director
name is acquisition).

the <application name> is the one that has been provided at the
application setup.
```

Output

The discovery will perform:

- The raw DFI discovery request
- Filter the discovered DFI with the appropriate filters
- Load all the discovered DFIs into the SQM Service Repository Manager (default load option: -diff no)

3.2.3.5 AMI directives

The following self-management commands are available using TIBCO Hawk Display User Interface (refer to the ***SQM Administration Guide*** where is explained how to use this console):

setTraceLogLevel, getTraceLogLevel setMtLogLevel, getMtLogLevel

As for all other OV SQM components

Dump

As for the other OV SQM components, the Dump method creates a Dump file in the trace files directory:

Argument : Dump Mode, can be one of the following:

- Config: the current configuration loaded in the module
- Memory: all the models and the current statuses
- Topics: the topics to which the module is subscribing
- All: all of the above (Config + Memory + Topics)

quietMode: stops the service adapter instance from publishing performance messages on the collection bus.

reloadConfig: prompts the service adapter instance to reload its configuration. This directive stops all data collection and re-activates them with the latest configuration data. The following application parameters can be reloaded using this directive:

- pollingPeriod (the minimum pollingPeriod is 0.5, which corresponds to 30 seconds)
- RequestRepliesNbRetry
- internalRequestRepliesTimeout

3.2.4 Starting / Stopping SA proxy

Starting and stopping an Service Adapter proxy application is done through the standard OV SQM management commands (described in the ***hp OpenView SQM Administration Guide***).

Prior to the stop and start commands, the user must:

- Connect as “sqmadm” user
- Load the OV SQM environment variables

The commands are as follows:

- Identify the SA proxy:

```
# temip_sc_show_director -platform slmv14 -director
acquisition -verbose

Processing
/tibco/private/adapter/ServiceCenter/PlatformDescription/slmv1
4/platform ...

Director acquisition :

      startid          = 20
      stopid           = 20
```

```
Application saproxy :
        applicationType  = Monitored
        host             = habine.vbe.cpqcorp.net
        command          = Proxy_v1_4_launch.sh
        startid          = 10
        stopid           = 20
        start_duration   = 1000
        stop_duration    = 20
        configpath       =
/tibco/private/adapter/ServiceCenter/ServiceAdapters/Proxy/v1_
4/saproxy

Application saproxy is NOT RUNNING

Director acquisition is NOT RUNNING

------------------------------------------------------------
--------

where:

   The SA Proxy application name is saproxy
```

- To start the application:

```
temip_sc_start_application -platform slmv14 -director
acquisition -application saproxy
```

- To stop the application:

```
temip_sc_stop_application -platform slmv14 -director
acquisition -application saproxy
```

3.3 Monitoring TeMIP Fault Service Adapter logs (TeMIP dedicated director)

The application logs (and internal tracing) are available at two different locations, as the TeMIP Fault Service Adapter uses two different Third Party Products:

- Tomcat, whose logs are available within $CATALINA_HOME/logs

The TeMIP Fault Service Adapter application logs are for the time being available together with the Tomcat logs within $CATALINA_HOME/logs. Indeed, remember that the Tomcat Web application container provides the whole execution environment, to run the TeMIP Fault Service Adapter application.

Chapter 4

How alarms impact services in OV SQM

The TeMIP Fault Service Adapter collects problem notifications (represented as TeMIP Alarm Objects) on the required TeMIP Operation Context. Alarm information is delivered as standard OV SQM measures. See the main OV SQM documentation for general information about measures.

4.1 The DFD

What follows is a description of the DFD; the actual DFD is available in the product as an XML-file.

DFD identifier

- DFD name = TeMIP_Fault_DFD
- DFD version = v1_0. The DFD version cannot be modified. The DFD version depends on the SA version
- DFD Label = TeMIP Fault Data Feeder Definition.

DFD properties

Properties part of the MRP: as the objective of the OV SQM TeMIP Fault SA, is to collect problem notifications on a given network resource, it is necessary to identify what the managed network resource is. As the OV TeMIP alarm collection is based on Operation Contexts (OC), the name of the OC is required to identify a problem instance.

The properties are:

- Network resource:
  - Name = NR
  - Label = Network Resource
  - Type = String
- Operation Context:
  - Name = OC
  - Label = Operation Context
  - Type = String

DFD Parameters

- Source Entity: contains the TeMIP Entity on which raw alarms occurred. This parameter will be used for drill-down on raw alarms (navigation from the SLA Monitoring to the TeMIP Client
  - Name = SourceEntity
  - Label = TeMIP Source Entity
  - Type = String. String format: <Full TeMIP Entity Name>, for instance: CiscoRouter north interface IfTable 7.
  - Default value: empty string

There are a number of Parameters that depend on which problem categories are defined.

The default TeMIP Fault Data Feeder Definition (by default) handles the following categories: CommunicationsAlarm, EnvironmentalAlarm, EquipmentAlarm, IntegrityViolation, OperationalViolation, PhysicalViolation, ProcessingErrorAlarm, QualityofServiceAlarm, SecurityServiceOrMechanismViolation, and TimeDomainViolation.

For each category, the following parameters are defined (All these parameters are customer independent):

- Problem Severity:
  - Name = <10 first characters of the category name>_Sever (special characters are removed from the category name)
  - Label = <category name> Severity
  - Type: enumeration
    - CLEARED (0)
    - WARNING (2)
    - INDETERMINATE (3)
    - MINOR (4)
    - MAJOR (5)
    - CRITICAL (6)
    - Default Value: CLEARED (0)
- Problem Alarm ID: TeMIP Alarm Object Identifier of the Problem Notification for the parameter category. This parameter will be used to drill-down on the problem notification (navigation from SLA Monitoring to TeMIP Client)
  - Name = <10 first characters of the category name>_PID
  - Label = <category name> Problem Alarm ID
  - Type = String. String format: (<OC Name>, <AO Id>)
  - Default Value: empty string
- Problem State: indicates the problem state.
  - Name = <10 first characters of the category name>_State
  - Label = <category name> Problem State
  - Type = Enumeration. Possible states are:
    - Outstanding (0)
    - Acknowledged (1)

- Terminated (2)
  Note: archived problems are considered as terminated
- Default Value: Terminated (2)

- Problem Creation Time: indicates the time the problem was created. This parameter can be used for computing the problem duration while the problem state is Outstanding.
  - Name = <10 first characters of the category name>_Time
  - Label = <category name> Problem Creation Time
  - Type = Absolute Time
  - Default value: null value
- Alarm information:
  - Name = <10 first characters of the category name>_Info
  - Label = <category name> Alarm Information
  - Type = String
  - Default Value = Empty String
- Custom parameters. The parameter value could be extracted from a user defined field of the TeMIP Alarm.
  - Name = <10 first characters of the category name>_<5 first characters of the parameter name>
  - Label = Defined in the parameter definition
  - Type = Defined in the parameter definition
  - Default Value = Defined in the parameter definition

4.2 The mapping

The following table describes how to map the information in a problem notification into a measure for OV SQM:

Measure parameter		Alarm Object attribute
DFD Identifier	Data Feeder Definition name	TeMIP_Fault_DFD
	Data Feeder Definition version	v1_0
Properties part of MRP	Network Resource	AO Managed Object field. (MO namespace is kept, the MO is converted in a string and set as property value)
	Operation Context	OC Name
Parameter	<Category Name> Severity	Severity
	<Category Name> State	State. ‘Archived’ state is here mapped into a ‘Terminated State’
	<Category Name> Creation Time	Alarm Creation timestamp: event time

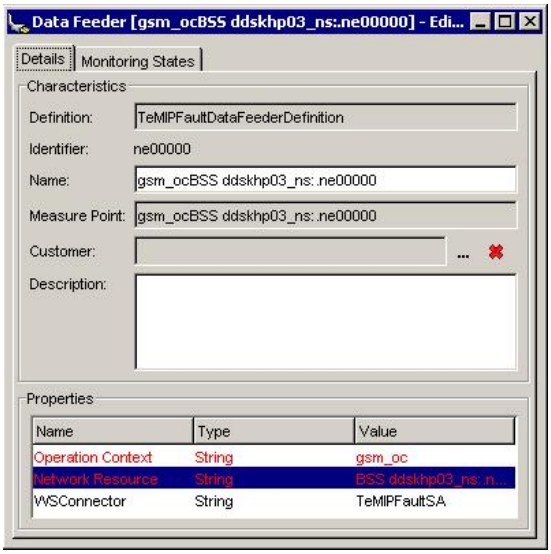
	<Category Name> Alarm Information	Additional Text
	Source Entity	The Service Adapter builds the parameter value from the AO custom field “Source Entity”
	<Category Name> Problem Alarm ID	(<OC Name>,<AO Id>) The OC name is the same as those specified in the MRP. It is duplicated here to avoid propagating the properties to the SCI and simplify UI Integration
	<additional parameters>	AO field having the same name as the DFD parameter
	Measure timestamp	Field AO event time
	Measure Identifier	Alarm Object Identifier
Measure’s specific information	isEventMeasure	True
	IsFinalrequestedMeasure	False
	IsOutage	Value of the events Outage field
	IsPartialMeasure	False
	IsRequestedMeasure	True
	IsStatusReport	False

4.3 Problem mapping example

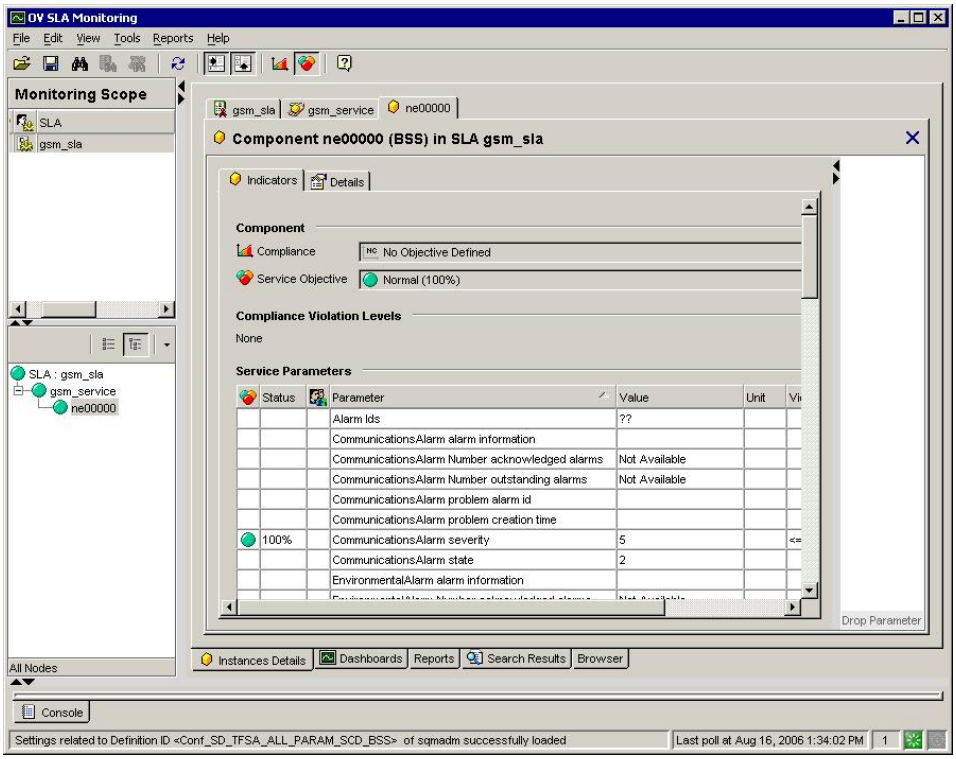
This example shows how a problem is mapped into a SCI in SQM.

TeMIP is configured to collect problems into an operation context with the name *gsm_oc*.

An SCI is declared to collect measures from operation context *gsm_oc* and network resource *BSS ddskhp03:.ne00000*.



When there is no active problem for an SCI a default measure is automatically generated for the unlocked SCI:



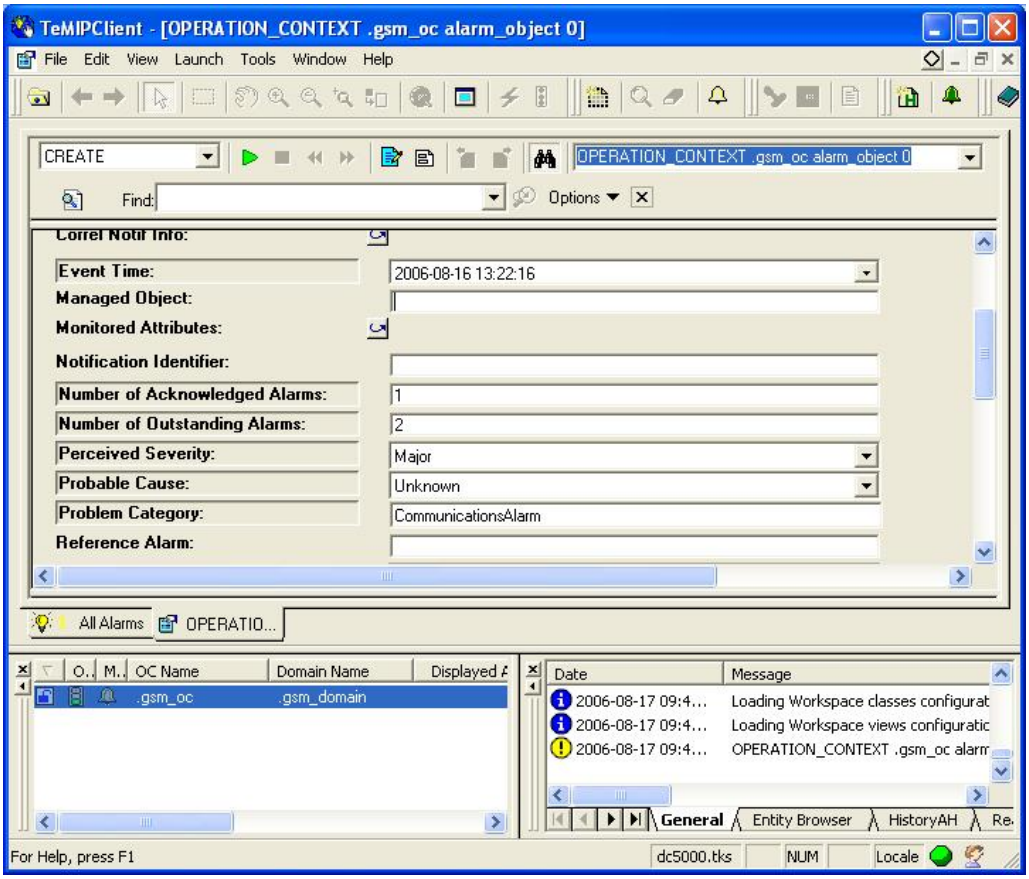
Default severity is Clear (5) and state is Terminated (2) for all problem categories.

The SCI parameters are affected by problem alarms collected by TeMIP.

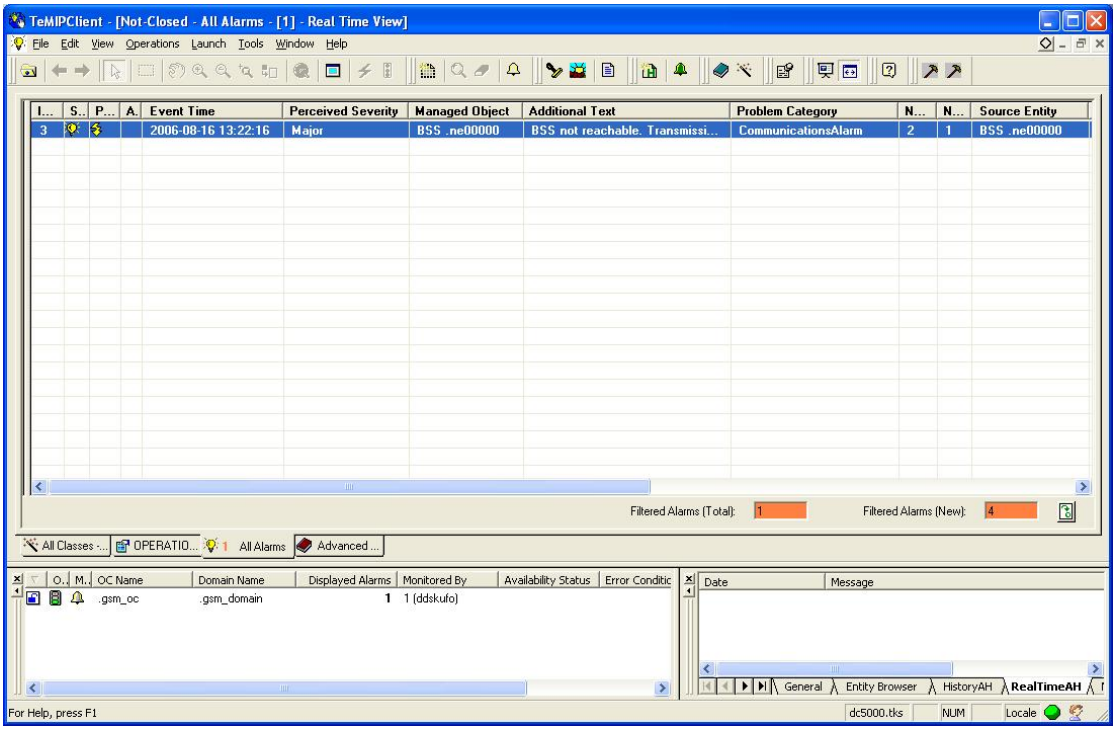
A problem alarm with all attributes set:

Attribute	Value
Identifier	3
Event Time	2006-08-16 13:22:16
Additional Text	BSS not reachable. Transmission adapter broken.
Source Entity	BSS .ne00000
Number of Outstanding Alarms	2
Number of Acknowledged Alarms	1
Perceived Severity	Major (2)
State	Outstanding (0)

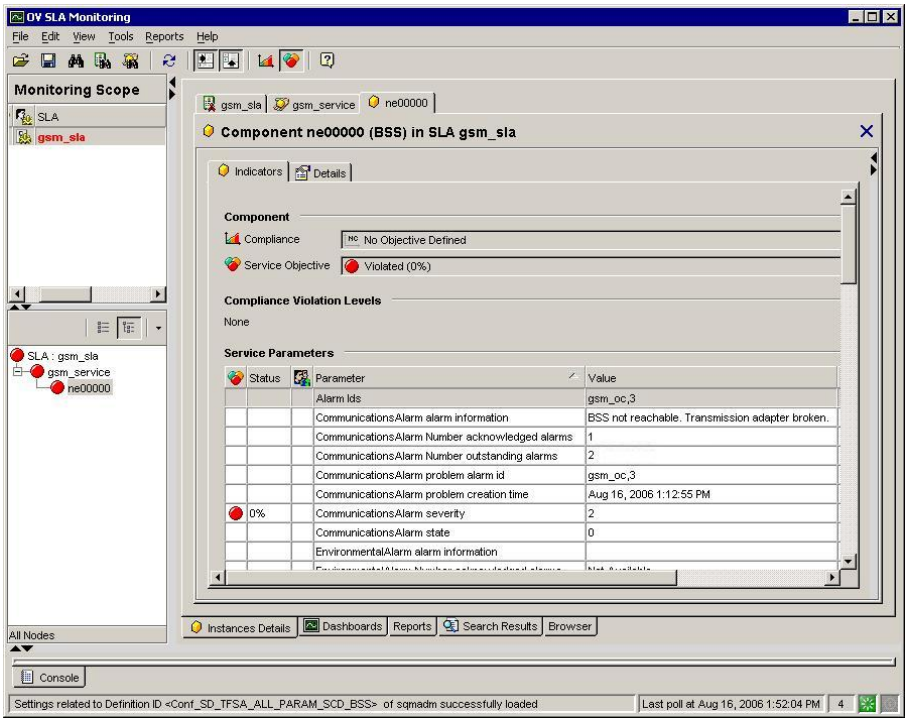
For test/demo purposes it is easiest to “create” problem alarms directly in the TeMIP client using the CREATE directive on the ALARM_OBJECT.



The created problem alarm shown in the TeMIP client:



The problem alarm mapped into the SCI in SQM with the impacted parameters:



This service level says that the SLA is violated when severity is 2 or lower.

Appendix A

DFI inventory file example

The DFI inventory file is used as input/output for each DFI discovery phase. Here is an example of inventory file, which syntax is important when customizing the filtering script.

```
<?xml version="1.0" encoding="UTF-8"?>
<inventory>
 <DFIEntry dfd.name="PerfDFD" dfd.version="v1_1"
  dfi.id="PerfDF_835227133" mrp.name="host1.vbe.cpqcorp.net"
  sa.name="PerfSA" sa.version="v1_1"    sai.id="slmv14_acquisition_myPerf"/>

 <DFIEntry dfd.name="PerfDFD" dfd.version="v1_1"
  dfi.id="PerfD__151287840" mrp.name="host2.vbe.cpqcorp.net"
  sa.name="PerfSA" sa.version="v1_1" sai.id="slmv14_acquisition_myPerf"/>

 <DFIEntry dfd.name="PerfDFD" dfd.version="v1_1"
  dfi.id="PerfDF_849885112" mrp.name="host3.vbe.cpqcorp.net"
  sa.name="PerfSA" sa.version="v1_1" sai.id="slmv14_acquisition_myPerf"/>
</inventory>
```

In the previous example, 3 DFIs have been discovered. Each DFI is identified by the tag **DFIEntry.** The DFI filtering script, is supposed to remove each entry that must not be loaded into OV SQM.

Appendix B

Filtering script example

The following example provides a DFI filtering program written in Perl language.

This program filters a raw discovery inventory file containing discovered DFI entries. The filtering is done on the MRP name: depending on the MRP name value, the DFI entry will be kept or not.

The output file is the Filtered inventory file.

To call the Perl program, the default filtering script has to be modified as follows:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application name>/<connector name>/discovery/filter/<platform name>_<director name>_<application name>_filter.sh`

```
#!/bin/sh
# Usage:
#  $1: raw file
#  $2: filtered file

RAWFILE=$1
FILTERFILE=$2

##
## Execute perl discovery filter

perl
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application
name>/<connector name>/discovery/filter/filter.pl -in $RAWFILE
-out $FILTERFILE

status=$?

echo "Filtering completed."

exit $status
```

Then the following Perl script has to be placed in the same directory as the filtering script:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application name>/<connector name>/discovery/filter/filter.pl`

```
use strict;
use Getopt::Long;
use XML::Simple;
##
## Constants
#######################
my $DFI_ENTRY_TAG = "DFIEntry";
my $MRP_NAME_ATTR = "mrp.name";
my $DFI_ID_ATTR = "dfi.id";
my $INVENTORY_ENTRY_TAG = "inventory";
main();
##
## filterInputDiscoveryFile
## Filter the input file on the MRP name value and put
the resulting
parsed XML into the specified output file
## Arguments:
##  inputDiscoveryFile : input XML file (raw discovery
file)
##  outputDiscoveryFile : output XML file (filtered
discovery file)
sub filterInputDiscoveryFile {
 my ($inputDiscoveryFile,$outputDiscoveryFile) = (@_);
 ## Check if the file exists
 ##  if yes, open it and parse it
 ## ============================
 if ( -f $inputDiscoveryFile ) {

  if ( -r $inputDiscoveryFile ) {
   ##
   ## Filtering consists in selecting DFIs where the MRP
name
contains 'MyString'
   ##

   my $xmlParser = new XML::Simple(keeproot => 1,
forcearray =>
['${DFI_ENTRY_TAG}']);
   my $inventory = $xmlParser-
>XMLin("${inputDiscoveryFile}");

   my $counter=0;

   # For each DFI Entry
   foreach my $dfiEntry ( @{$inventory-
>{"${INVENTORY_ENTRY_TAG}"}-
>{"${DFI_ENTRY_TAG}"}}) {
      my $dfiID=${dfiEntry}->{"${DFI_ID_ATTR}"};
      $_=${dfiEntry}->{"${MRP_NAME_ATTR}"};
      if ( /MyString/ ) {
       # The MRP Name matches the keyword 'MyString' so
keep this
DFI
       print "$dfiID is kept\n";
      } else {
       # The MRP Name does NOT match the keyword
'MyString' so
delete this DFI
       print "$dfiID is filtered-out\n";
       delete $inventory->{"${INVENTORY_ENTRY_TAG}"}-
>{"${DFI_ENTRY_TAG}"}[$counter];
```

```
      }
      $counter++;
   }
   # Generate the filtered Discovery file
   XMLout($inventory,keeproot => 1 , suppressempty =>
1,keyattr =>
['${DFI_ENTRY_TAG}'], outputfile => $outputDiscoveryFile
);
# Hack: re-parse the filtered file to remove empty
values and
regenerate the output file
   my $xmlParser2 = new XML::Simple(keeproot => 1,
suppressempty =>
1,forcearray => ['${DFI_ENTRY_TAG}']);
   my $inventory2 = $xmlParser2-
>XMLin("${outputDiscoveryFile}");
   XMLout($inventory2,keeproot => 1 , suppressempty =>
1,keyattr =>
['${DFI_ENTRY_TAG}'], outputfile => $outputDiscoveryFile
);

  } else {
   print ("Warning: cannot read file:
${inputDiscoveryFile}\n");
  }
 } else {
  print ("Warning: cannot find file:
${inputDiscoveryFile}\n");
 }
}
############################################################
#############
##########
# Main
#
# arguments:
#  -in <file> : raw discovery file
#  -out <file> : filtered discovery file
############################################################
#############
##########
sub main {
 my $inputFile;
 my $outputFile;
 my $optStatus=&GetOptions('in=s'  => \$inputFile,
                   'out=s'  => \$outputFile);

 if ( !$optStatus ) {
  print ("ERROR: invalid option \n");
  exit 2;
 }
 filterInputDiscoveryFile($inputFile,$outputFile);
}
```

Appendix C

Troubleshooting

TeMIP Service Adapter Troubleshooting

To enable tracing facilities, set required trace information by supplying a file containing the appropriate trace flags to /optOV/support/ovtrccfg:

```
TCF Version 3.2
APP: "TFSA"
SINK: Socket "127.0.0.1" "node=127.0.0.1;"
TRACE: "tfsa" "Trace" Info Warn Error Developer Verbose
TRACE: "service_api" "Trace" Info Warn Error Developer Verbose
TRACE: "servlet" "Trace" Info Warn Error Developer Verbose
TRACE: "temip" "Trace" Info Warn Error Developer Verbose
TRACE: "context" "Trace" Info Warn Error Developer Verbose
TRACE: "measures" "Trace" Info Warn Error Developer Verbose
```

It depends on how XPL is configured where these traces will end up.

The following table describes the valid component ids:

Component id	Description
context	Logging at context level. • Registration/Deregistration. • Context related measure collection.
measures	Measure content logging.
service_api	Logging at SOAP interface level.
servlet	Logging at Servlet level. • Startup/Shutdown of the TeMIP Fault service adapter. • Reading of configuration files.
temip	TeMIP related logging.
tfsa	Overall tfsa logging.

Proxy Service Adapter Troubleshooting

The SA Proxy logging and tracing is done in the TEMIP_SC_VAR_HOME directory if this variable was defined at the SA proxy setup. Otherwise, the traces and logs are redirected into the directory provided at the setup:

```
TEMIP_SC_VAR_HOME/log
TEMIP_SC_VAR_HOME/trace
```

The files are identified as follows:

```
<platform>_<director>_<application>.log
```

To enable Proxy Service Adapter tracing facilities, set required trace information by updating the application configuration file located in:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/config/<platform>_<director>_<application>.properties

For the SA proxy application, as for other OV SQM components, you can refer the *HP OpenView Service Quality Manager Administration Guide* for troubleshooting information.

Discovery tool Troubleshooting

The SA Proxy discovery tool (temip_sc_discovery.sh) logging and tracing is done at the following location:

```
$TEMIP_SC_VAR_HOME/log
$TEMIP_SC_VAR_HOME/trace
```

The files are identified as follows:

```
SQM_Proxy_v1_4_<application>_Discovery.log
```

To enable discovery tracing facilities, set required trace information by updating the application configuration file located in:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_4/<application>/config/SaProxyDiscoveryTraceLogging.properties
```

To enable all levels of trace set the property named '.level' to 'ALL.

This property file defines also the location of the trace file thank to the variable named 'com.compaq.temip.servicecenter.common.logging.FileHandler.pattern'

Appendix D

Acronyms

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

Term	Description
API	Application programming interface
DFD	Data feeder definition
DF	Data feeder = Data feeder instance
MRP	Measurement reference point
SAI	Service Adapter Application Name (or Service Adapter instance)
SCI	Service Component Instance
SLA	Service level agreement
SLM	Service level management
SLO	Service level objective
SRM	Service Repository Manager
XML	eXtensible Mark-up Language


invent


Java™
COMPATIBLE