HP OpenView Service Quality Manager

SA proxy

Installation, Configuration and User Guide

Edition: 1.4

March 2007

Legal Notices

Contents

Preface

This document describes how to install and use the hp OpenView Service Quality Manager (SQM) Service Adapter proxy. The Service Adapter proxy application, once configured, is able to interoperate with a Service Adapter (Web Services) in order to configure it, to discover the Data Feeder Definitions (DFD) and Instances (DFI) it manages, to control and finally to retrieve the quality of service measures of the these DFIs.

This document describes how to:

- Use Service Adapter proxy applications
- Install and setup a Service Adapter proxy application

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 the chapter 2.1.1

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

SA Topology

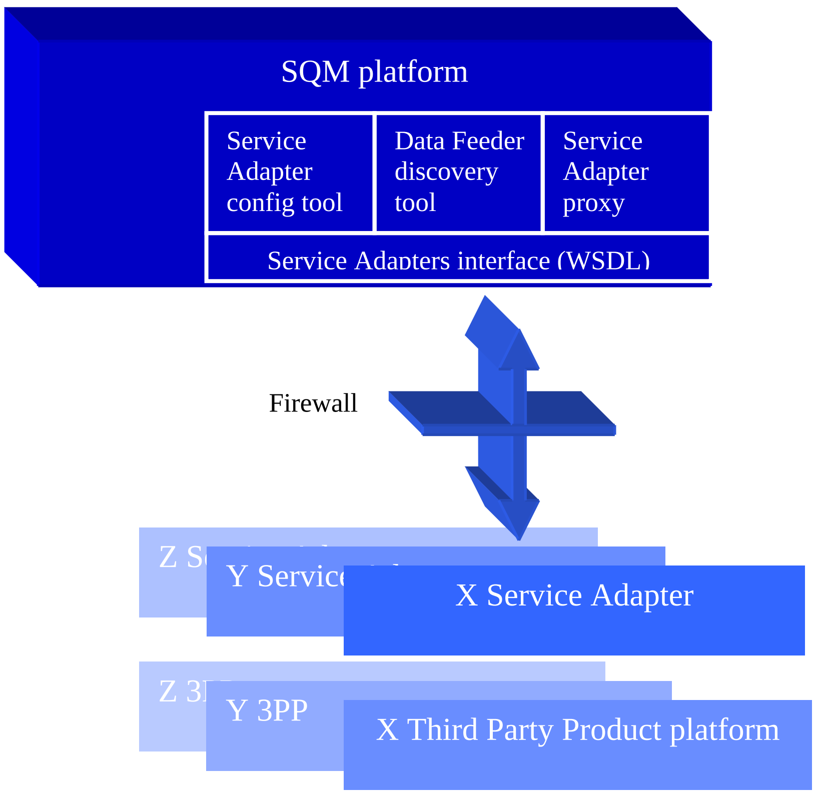
1.1 Platform view

The SQM platform southbound is today composed of various data sources integration modules. The integration modules are specific to a category of data source. SQM provides various integration modules, like for example an OVIS integration module for OVIS data sources, or a TeMIP integration module for TeMIP data sources. A module also provides the data model, describing the data source. This data model is required by the SQM platform to be able to perform collections on the data source.

The Web Service based Service Adapters (SA) whereas belong to a different category of data sources. Indeed, these data sources, opposed to the OVIS or TeMIP data source, could be configured, discovered, controlled and collected through a common and standard interface, the Service Adapter interface (WSDL). Therefore, only one single integration module, the SA Proxy, is used to interoperate with this new kind of data sources, called the Service Adapters.

The schema below represents, at the bottom the various possible Service Adapters and at the top the SQM platform including the dedicated integration module: a SA proxy.

Indeed, the SA proxy is decomposed into a Service Adapter configuration tool, a Data Feeder discovery tool and the Service Adapter proxy it self. Refer to the [SQM Overview] and [SQM Model] for further details on the Data Feeder terminology.


SQM platform
Service Adapter config tool
Data Feeder discovery tool
Service Adapter proxy
Service Adapters interface (WSDL)
Firewall
Z S
Y Service
X Service Adapter
Z 3
Y 3PP
X Third Party Product platform

1.2 Software component view

The generic Service Adapter proxy is a software component part of SQM, which has to be installed on the SQM platform.

A Service Adapter whereas is most likely installed on the platform that runs the Third Party Product that the Service Adapter has to wrap.

To run a SA Proxy connecting to a given Service Adapter, the SA Proxy software has first to be installed (confer 2.1 Installing a SA proxy) and then set-up accordingly to the Service Adapter(s) (X, Y, Z and/or T) it should connect to. Note that the SA proxy has to be installed only once. Indeed, the SA Proxy setup effectively creates a SA Proxy application, also known as Service Adapter Instance (SAI). Although the SA proxy applications run independently from each other (different processes), they share the same SA proxy software binaries.

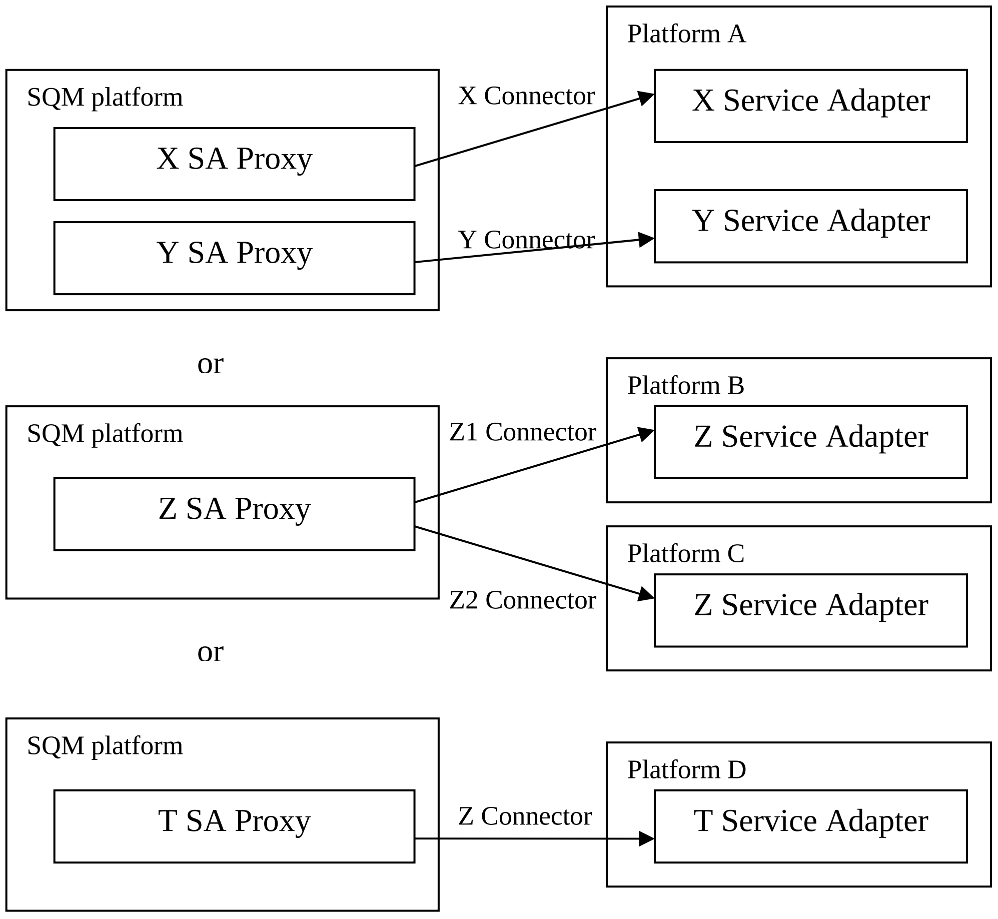
The SA proxy setup is composed of the following steps:

- SA proxy application (SAI) creation
- SA proxy (connector) configuration
- Service Adapter and Third Party Product configuration
- Data Feeder model (DFD and DFI) discovery per Service Adapter

1.3 SA proxy – SA interoperability

The following schema describes the possible interconnections between the SA proxy applications and given Service Adapters it could collect from. Note that the SA proxy applications use different (named) connectors to interoperate with their associated Service Adapters.

Indeed, Service Adapters are visible as Web Services that implement the standard SA interface, delivered by the SQM SA SDK. The SA proxy, like any other web proxy, could interoperate with these Service Adapter Web Services, using an URL. This URL contains the host name, the port number, as well as the name of the Service Adapter which is available at this location.



1.4 Preliminary “design” phase

Please take a while before setting up a SA Proxy application! Indeed, it is important to know exactly how many and which Service Adapters you would like to integrate into SQM.

As exposed in 1.3 SA proxy – SA interoperability, the SA proxy application could interoperate with one or several Service Adapters. You have to choose the SA Proxy / Service Adapter (Third Party Product) combination that will match the requirements.

Let us take the Z SA Proxy as an example. We have two platforms B and C which both run a Z Third Party Product, like for example TeMIP. So we would like to setup a TeMIP SA Proxy, which interoperates with these both TeMIP platforms. First of all we attribute a unique name to the SA proxy application, lets say: Corporate_TeMIP_SA_Proxy. This proxy will interoperate, i.e. connect to both a France subsidiary TeMIP Service Adatper and a Germany subsidiary TeMIP Service Adapter. It is important to attribute a meaningful name to these SA connectors, like France_TeMIP_SA and Germany_TeMIP_SA. These connector names are required during the Corporate_TeMIP_SA_Proxy setup and later upon the Data Feeder discovery.

In the following chapter, the SA proxy application name will be designated through <Application name>, and the name (id) of the Service Adapter dedicated connectors through <Connector name>.

In theory, the <Connector name> could look like hotel.cpqcorp.net_8080_TeMIP_SA (a concatenation of the Service Adapter host name followed by its port and finally the Service Adapter name). But following the best practice, it is safer to use a name like France_TeMIP_SA, as the Service Adapter’s host and port will most likely change over time.

Chapter 2

SA proxy installation and configuration

2.1 Installing a SA proxy

2.1.1 Software requirements

As for the SA proxy, the SA proxy kit requires that the following software:

- HP-UX V11.11
- HP OpenView Service Quality Manager V1.2 (Kernel subset)
- HP OpenView SA Common V1.21 (**SQMSACOMMON**)

Prior to the SA Proxy installation, the SQM Kernel and the Service Adapter Common components have to be installed.

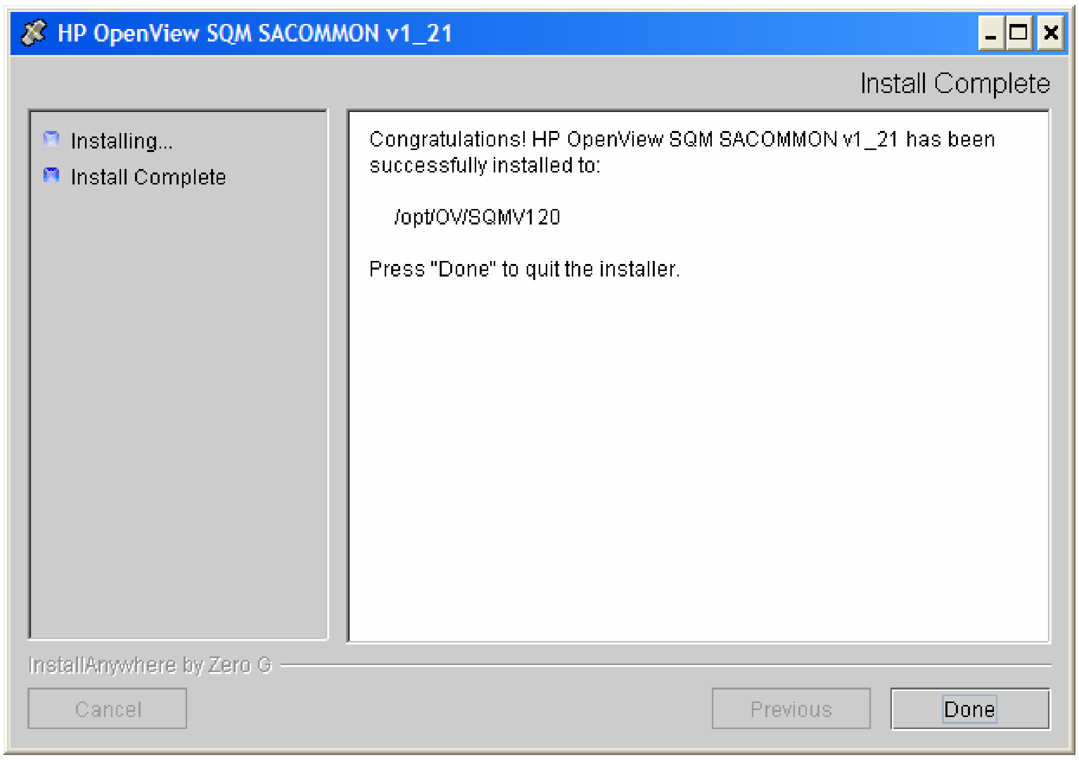
2.1.2 Installing the SA Common subset

If necessary, install the SA Common component by doing the following. If this has already been done, go directly to "**Error! Reference source not found.**" on page **Error! Bookmark not defined.**.

1. First, log on to the system as **root** user.
2. Mount the HP OpenView Service Adapters and Gateways CD-ROM on your system.
3. Go to <*mount directory*>/SQM-1.20.00
   and execute the following command:

```
# ./SQMSACOMMON-1.21.00.bin
```

4. The software is installed and the **Install Complete** window is displayed.



Congratulations! HP OpenView SQM SACOMMON v1_21 has been successfully installed to:

/opt/OV/SQMV120

Press "Done" to quit the installer.

2.1.3 Installing on HP-UX

On HP-UX, follow the below listed steps to install the SA proxy kit.

- 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-1.20.00
- Set the TEMIP_SC_HOME environment variable to the SQM Root directory:

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

- Install the Service Adapter proxy InstallAnyWhere kit.

```
# SQMSAPROXY-1.20.00.bin
```

2.1.4 Configuring the SQM Kernel

The SQM Kernel needs to be setup to run a SA proxy. Three different setups apply, depending on the platform environment:

5. The SA proxy is installed on the HP-UX SQM SLM Primary Server:

   In this case, please refer to ***hp Openview SQM Installation Guide*** to perform the setup of the SQM SLM Primary Server.

6. The SA proxy is installed on a HP-UX system distinct from the SQM SLM Primary Server where the SQM Kernel has not been configured:

   In this case, it is necessary to retrieve the SLM Server platform description file:

- Create the **sqmadm** administration user on the targeted Unix system (refer to the *hp Openview SQM Installation* Guide for the user account creation)

a. Retrieve the file ***$TEMIP_SC_VAR_HOME/setupconfig/platform_desc.cfg*** from the SQM SLM Primary Server, and copy it on the SA proxy HP-UX system in ***$TEMIP_SC_HOME/tmp***

b. Connect as **root** user to run the following commands:

```
# export TEMIP_SC_HOME=<SQM installation
directory>
# cd $TEMIP SC HOME/setup/bin
# temip_sc_setup -all -NI
```

2.2 Setting up a SA proxy application

Before setting up a SA proxy application

Please read carefully paragraph 1.4 Preliminary “design” phase.

General processing

Important Note

Before the SA proxy setup, it is mandatory that the SQM Kernel setup has been performed (see previous chapter).

The setup and configuration of the Service Adapter proxy is done in 2 steps:

- The application setup step, which declares the SA proxy application into the SQM Central Repository and creates the SA proxy application data tree within the $TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/ location.
- The creation of the connectors to the remotes Service Adaptors. This configuration step, which prompts the user for the Service Adapter Web Service host name, host port and SA name. This configuration data is not only loaded into the SQM Central Repository, but is also used to add sub-directories within the SA proxy application data tree.

2.2.1 Configuring on HP-UX

The setup tool is located in:

`$TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin`

2.2.1.1 Application setup

CAUTION: Remember that the SA proxy application names have to be unique cross platform and director!

The following steps have to be performed to setup and configure the application:

Application creation

This phase consists in creating a SA proxy application on the SQM platform (on a specified director).

1) Check the kernel is running: the temip_sc_setup command stops the kernel at the end of the process. If kernel is not started, perform the following command:

```
# temip_sc_kernel_start
```

2) Check if SRM is running. If not, start the application by performing the following command:

```
# temip_sc_start_application -platform <platform_name> -director
slmonitoring -application SRM
----------------------------------------------------------------------

   where:

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

Command:

- Connect as "**root**" user.
- Load the SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)
- Perform the following commands to create a SA Proxy Application:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

# temip_sc_configure.sh -dirName <director name> -applicationName
<application name> -setup
----------------------------------------------------------------------

   where:

    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. Confer the SA proxy name.
```

The application name has to be provided by the user.

Output:

This command creates a Service Adapter proxy application, by updating SQM Central Repository and a adding an application data tree at the following location.

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/<Application name>`

This location contains:

- config/SaProxyDiscoveryMtLogging.properties: Discovery tool logging properties file
- config/SaProxyDiscoveryTraceLogging.properties: Discovery tool tracing properties file
- config/*<director name>*_*<platform name>*_*<application name>*.properties: SA Proxy Application tracing and logging property file
- repository/

```
----------------------------------------------------------------------

   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. See <SA proxy name>
```

2.2.1.2 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 Adapter (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 SQM Central Repository.

Associate a new Service Adapter (create a connector)

Command:

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

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

# temip_sc_configure.sh -applicationName <application 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.
```

This command will prompt the user for Service Adapter (Web Service) location parameters: SA host name (including the domain name), Web Service port number, SA name (for example SampleSA_v1_0). The command not only loads these parameters into the SQM Central Repository, but also uses the connector name to extend the application data tree as follow:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/<Application
name>/<Connector name>
```

This location contains:

- SAConfig/: remote Service Adapter configuration file (reserved for future use)
- discovery/filter/: directory containing the discovery DFI filtering script
- discovery/inventory/: discovery directory
- discovery/inventory/raw/: directory containing the raw discovery information
- discovery/inventory/filtered/: directory containing the discovery information after filering
- discovery/repository/: directory containing DFD and DFI XML files (resulting of the discovery phase) and also Tibco repository backup files

Output (*including interactive prompts*):

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

Create the Connector datatree.

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>   (created)

/var/opt/OV/SQM OL V120/slmv12/ServiceAdapters/Proxy/v1 2/<applica
tion name>/<connector name>/SAConfig   (created)

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>/discovery   (created)

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>/discovery/filter   (created)

/var/opt/OV/SQM OL V120/slmv12/ServiceAdapters/Proxy/v1 2/<applica
tion name>/<connector
name>/discovery/filter/slmv12_acquisition_SAProxy_OL_filter.sh
(created)

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>/discovery/inventory   (created)

/var/opt/OV/SQM OL V120/slmv12/ServiceAdapters/Proxy/v1 2/<applica
tion name>/<connector name>/discovery/inventory/raw   (created)

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>/discovery/inventory/filtered
(created)

/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/<connector name>/discovery/repository   (created)

Warning: parameter config file
/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/config/<connector name>.cfg not found

Please enter the Service Adapter (Web Service) Name: SampleSA_v1_0

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

Please enter the Service Adapter (Web Service) Port Number: 1973

Load the Connector in the Tibco Repository

INFO: Backup written at the following location:
/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/repository.2005_4_28_17_58_51

INFO:
/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/repository/connectors_data.exp has been imported into
the Repository

INFO: Backup written at the following location:
/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/repository.2005_4_28_17_59_08

INFO:
/var/opt/OV/SQM_OL_V120/slmv12/ServiceAdapters/Proxy/v1_2/<applica
tion name>/repository/monitored_connectors_data.exp has been
imported into the Repository

Add Connector succeed.
```

2.2.1.3 List all associated Service Adapters (list all connectors)

Command:

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

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

# temip_sc_configure.sh -applicationName <application name> -
listConnectors

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

where <application name> is the application name provided at the setup
command.
```

This command lists all the connectors (name and parameters) that are available within the SQM Central Repository for the given SA Proxy application.

Output example:

```
<InternalReference>
    <Hard_SampleSA_v1_0>
      <RepairDFServiceInterval>60000</RepairDFServiceInterval>
      <HostName>hard.vbe.cpqcorp.net</HostName>
      <SAName>SampleSA_v1_0</SAName>

<SAProvidesAtLeast1DFMeasureAfterTimeout>50000</SAProvidesAtLeast1DFMeasureAfterTimeout>
      <NbRetryOnConnectionFailure>3</NbRetryOnConnectionFailure>
      <URL>http://__HostName__:__PortNumber__/__SAName__/services</URL>
      <PortNumber>1973</PortNumber>
      <StartDFServiceInterval>30000</StartDFServiceInterval>
      <StopDFServiceInterval>30000</StopDFServiceInterval>

<RetrieveDFMeasuresServiceRetryInterval>30000</RetrieveDFMeasuresServiceRetryInterval>
      <RetrieveDFMeasuresBundlesMaxSize>150</RetrieveDFMeasuresBundlesMaxSize>
      <ControlDFBundlesMaxSize>150</ControlDFBundlesMaxSize>
      <ServiceOperationsTimeout>60000</ServiceOperationsTimeout>
    </Hard_SampleSA_v1_0>
</InternalReference>
```

2.2.1.4 Disassociate a Service Adapter (remove a connector)

Command:

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

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

# temip_sc_configure.sh -applicationName <application name> -
removeConnector <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.
```

This command removes the specified connector from the SQM Central Repository. Note that associated DFDs are not removed from the Service Adapter Proxy Application repository. Use the next command (2.2.1.5) to dissociate a monitored DFD.

2.2.1.5 Disassociate a monitored DFD

Command:

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

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

# temip_sc_configure.sh -applicationName <application name> -removedfd -
dfdName <dfd name> -dfdv <dfd version>

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

   where:

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

    the <dfd name> identifies the Data Feeder Definition to dissociate

    the <dfd version> is the Data Feeder Definition version to
dissociate.
```

This command dissociates the specified Data Feeder Definition from the given SA Proxy Application. Once dissociated, the DFD won't be any longer monitored by the application. Nevertheless, the DFD is not removed from the SRM (use the command temip_sc_delete_dfd to remove a DFD from the SRM)

2.2.1.6 Checking connector availability

Command:

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

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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:

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

2.3 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 be automatically loaded in the SQM Service Repository Manager.

Discovery script

The discovery script is located in the following directory:

`$TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin/temip_sc_discover y.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 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 **One shot discovery and loading** for more details on this command.

2.3.1 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 SQM environment variables

 (default: /var/opt/OV/SQM/slmv11/temip_sc_env.sh)

- Perform the following commands:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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_2***<application name>/<connectorname>***/discovery/repository/**NewDFDReq_<DFDName>.<DFDVersion>**.xml

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2***<application name>/<connectorname>***/discovery/repository/**DelDFDReq_<DFDName>.<DFDVersion>**.xml

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

Note

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

2.3.2 Loading phase

Command

The discovery loading request has to be performed as follows:

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

(default: /var/opt/OV/SQM/slmv11/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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 SQM Service Repository Manager the Data Feeder Definition (XML files) located in:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2<***application name***>/<***connectorname***>/discovery/repository/

2.3.3 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 SQM environment variables

(default: /var/opt/OV/SQM/slmv11/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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 SQM Service Repository Manager

2.4 Discovering and loading Data Feeder Instances (DFIs)

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

Important Note

Before performing discovery phases, it is mandatory to setup a SA Proxy application (see chapter 2.2.1.1) and to create a connector on a remote Service Adapter (see chapter 2.2.1.2).

Discovery script

The discovery script is located in the following directory:

```
$TEMIP_SC_HOME/ServiceAdapters/SaProxy/v1_2/bin/temip_sc_discov
ery.sh
```

Script Usage

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

The discovery parameters:

- -connectorName: 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: performs the discovery phase only
- -filter: performs the discovery filtering phase only.
- –load: performs the loading phase only: discovered DFDs 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: 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 reffile

  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 Instance 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 srm

  This option performs the same Data Feeder Instances comparisons as the ***reffile*** mode, but instead of considering a reference file, the declaration will depend on the existence of the Data Feeder Instance in 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 filtered Data Feeder Instances: **-diff no**). Please refer to chapter **One shot discovery and loading** for more details on this command.

2.4.1 Raw discovery phase

This initial phase will retrieve the DFI 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/slmv11/temip_sc_env.sh)

- Perform the following commands:

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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_2/<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_2/<application name>/<connectorname>/discovery/repository/DeclareDFIReq_<DFDName>.<DFDversion>.<DFIID>.xml`

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2<application name>/<connectorname>/discovery/repository/DelDFIReq_<DFDName>.<DFDversion>.<DFIID>.xml`

2.4.2 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 that should not be managed by the SA proxy application.

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

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

By default, a filtering script is provided with the SA proxy, 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:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/<application name>/<connector name>/discovery/filter/<platform name>_<director name>_<application name>_filter.sh`

Note

The filtering script can be processed 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 C.

- The raw DFI inventory file is located at:

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/`***`<application name>/<connector name>`***`/discovery/inventory/raw/`***`<platform name>_<director name>_<application name>`***`.xml`

Command

The discovery filtering request has to be performed as follows:

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

 (default: /var/opt/OV/SQM/slmv11/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

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

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

   where:

    the <connector name> of the connector that designates the Service
Adapter for which the Data Feeder Instances have to be filtered. 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

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

`$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/`***`<application name>/<connectorname>`***`/discovery/inventory/filtered/`***`<platform name>_<director name>_<application name>`***`.xml`

2.4.3 Loading phase

This phase loaded the DFIs defined in the filetered inventory file into the SQM SRM.

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 reffile**

This option will compare the list of discovered/filtered Data Feeder Instances against a DFI reference file.

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 Instance 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 srm**

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

Input

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

It is available at:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/***<application name>/<connectorname>***/discovery/inventory/filtered/***<platform name>_<director name>_<application name>***.xml

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/v1_2/***<application name>/<connector name>***/discovery/repository/***<platform name>_<director name>_<application name>***_discovery_reference.xml

Command

The discovery loading request has to be performed as follows:

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

(default: /var/opt/OV/SQM/slmv11/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/bin

#temip_sc_discovery.sh -platform <platform name> -director <director
name> -application <application name> -connector <connector name> -dfi -
load [ -diff (no | reffile | srm)]



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

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

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

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/***<applicationname>/*** ***<connector name>***/discovery/repository***/<platformname>***_ ***<directorname>***_***<applicationname>***_discovery_cmds.log

$TEMIP_SC_VAR_HOME/ServiceAdapters/Proxy/***<applicationname>/*** ***<connector name>***/discovery/repository/***<platformname>***_ ***<director name>***_***<applicationname>***_discovery_cmds.sh

2.4.4 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/slmv11/temip_sc_env.sh)

- Perform the following commands

```
# cd $TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/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**)

2.4.5 Scheduling the DFI discovery

It is recommended to encapsulate all the previously DFI discovery commands into specific scripts that can run in a crontab.

The discovery will be run in batch mode, to load automatically newly discovered DFIs from the remoter Service Adapters.

2.4.6 DFI discovery advanced configuration

This section describes some configuration parameters of the discovery tool (temip_sc_discovery application).

The application configuration file is located in:

$TEMIP_SC_HOME/ServiceAdapters/Proxy/v1_2/properties/SaProxyDiscovery.properties.

Configuration parameters allow controlling Data feeder loading in SQM SRM. There are 2 available parameters:

- **discovery.dfi.max_dfi_per_message**: this parameter defines the maximum number of DFI declarations is a single message sent to the SRM. Default value: 10000. If the number of messages exceeds this limit, the discovery tool splits in multiple messages.
- **discovery.dfi.srm_timeout_in_sec**: timeout in seconds when sending DFI declaration request to the SRM. Default value=600. When there is an important number of a DFI declaration, it is recommended to increase this value. The number of DFI declaration per message can also be reduced to avoid timeout.

Note

It is not recommended to modify other configuration parameters.

2.5 Advanced application configuration

This chapter describes how to setup the Service Adapter proxy application configuration variables and the available application self-management directives. To access these capabilities, it is necessary to edit the SQM Central Repository using the Tibco Designer or start the Tibco Hawk Display console. These applications are described in the *hp OpenView SQM Administration Guide*. Please refer to this document for more information about these administration tools.

2.5.1 Application and connection configuration variables

For Service Adapter proxy application advanced configuration, the user may open the SQM Central Repository and edit the following URL using the TIBCO Designer (see ***SQM Administration Guide***):

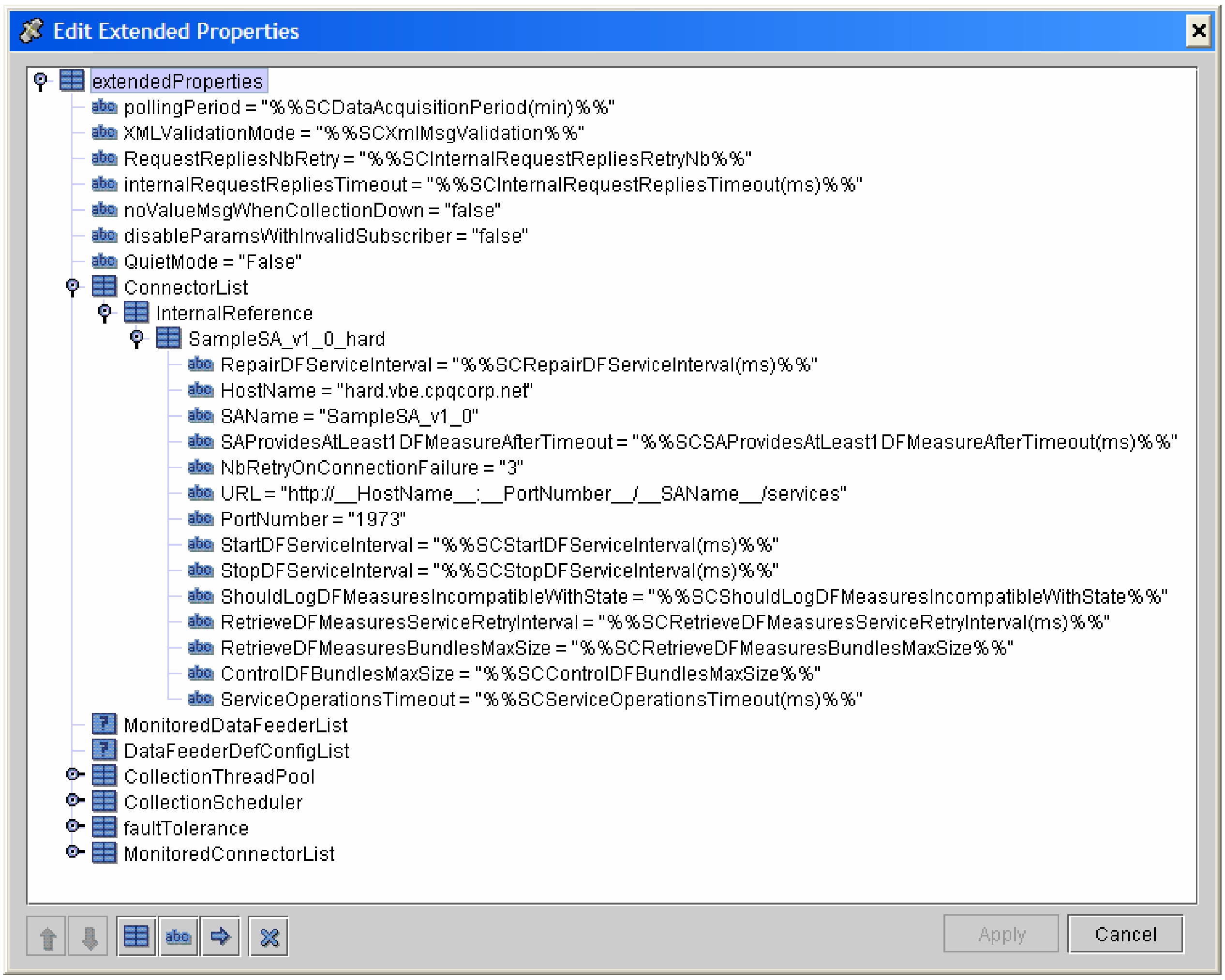
/tibco/private/adapter/ServiceCenter/ServiceAdapters/ServiceAdapters/SaProxy/v1_2 /<**applicationname**>_config/ConnectorList/InternalReference/<**Connector name**>

A general description of the Service Adapters application configuration variables is available in the ***hp OpenView SQM Administration Guide***. The following section will describe only the variables that influence the SA proxy behavior.

Variable Name	**Default**	**Description**
`SAName`	n/a	SA (Web Service) name (eventually including the SA version)
`HostName`	n/a	SA (Web Service) hostname (including the domain)
`PortNumber`	n/a	SA (Web Service port on HTTP server)
`URL`	http://__HostName__:__ PortNumber__ /__SAName__/services	SA (Web Service) URL

The URL variable defines a default URL pattern composed of the values of the SAName, HostName and PortNumber variables. Note that the leading and ending "__" strings should not be removed, as these are mandatory to allow variable substitutions. Of course, the URL could be changed to other valid URL. In most cases it is not necessary to modify this variable.

Here after a screenshot of the TIBCO Designer editing the /tibco/private/adapter/ServiceCenter/ServiceAdapters/ServiceAdapters/SaProxy/v1_2 /<**applicationname**>_config/ConnectorList/InternalReference/<**Connector name**>


```
extendedProperties
  pollingPeriod = "%%SCDataAcquisitionPeriod(min)%%"
  XMLValidationMode = "%%SCXmlMsgValidation%%"
  RequestRepliesNbRetry = "%%SCInternalRequestRepliesRetryNb%%"
  internalRequestRepliesTimeout = "%%SCInternalRequestRepliesTimeout(ms)%%"
  noValueMsgWhenCollectionDown = "false"
  disableParamsWithInvalidSubscriber = "false"
  QuietMode = "False"
  ConnectorList
    InternalReference
      SampleSA_v1_0_hard
        RepairDFServiceInterval = "%%SCRepairDFServiceInterval(ms)%%"
        HostName = "hard.vbe.cpqcorp.net"
        SAName = "SampleSA_v1_0"
        SAProvidesAtLeast1DFMeasureAfterTimeout = "%%SCSAProvidesAtLeast1DFMeasureAfterTimeout(ms)%%"
        NbRetryOnConnectionFailure = "3"
        URL = "http://__HostName__:__PortNumber__/__SAName__/services"
        PortNumber = "1973"
        StartDFServiceInterval = "%%SCStartDFServiceInterval(ms)%%"
        StopDFServiceInterval = "%%SCStopDFServiceInterval(ms)%%"
        ShouldLogDFMeasuresIncompatibleWithState = "%%SCShouldLogDFMeasuresIncompatibleWithState%%"
        RetrieveDFMeasuresServiceRetryInterval = "%%SCRetrieveDFMeasuresServiceRetryInterval(ms)%%"
        RetrieveDFMeasuresBundlesMaxSize = "%%SCRetrieveDFMeasuresBundlesMaxSize%%"
        ControlDFBundlesMaxSize = "%%SCControlDFBundlesMaxSize%%"
        ServiceOperationsTimeout = "%%SCServiceOperationsTimeout(ms)%%"
  MonitoredDataFeederList
  DataFeederDefConfigList
  CollectionThreadPool
  CollectionScheduler
  faultTolerance
  MonitoredConnectorList
```

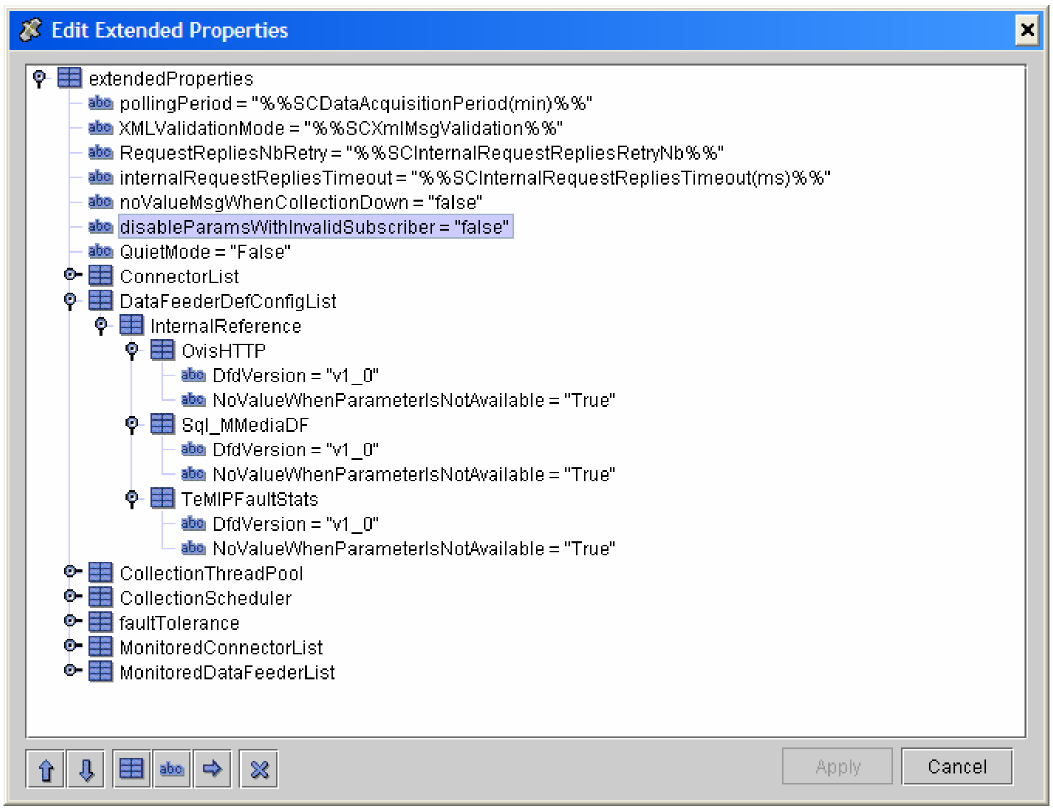
In the **ConnectorList** variable lists the connection parameters used to access the different Service Adapters, as well as the following parameters that could be configured:

Variable Name	Default	Description
NbRetryOnConnectionFailure	3	
StartDFServiceInterval	%%SCStartDF ServiceInterval (ms)%% Global variable 10000 ms	Elapsed time (in milliseconds) before propagating a DFI unlock request to the remote Service Adapter. This time interval allows sending unlock requests in batch mode to the service adapter.
StopDFServiceInterval	%%SCStopDF ServiceInterval (ms)%% Global variable 10000 ms	Elapsed time (in milliseconds) before propagating a DFI lock request to the remote Service Adapter. This time interval allows sending lock requests in batch mode to the service adapter.

RepairDFServiceInterval	%%SCRepairDFServiceInterval(ms)%% Global variable 15000 ms	Elapsed time (in milliseconds) before propagating a repair request to the remote Service Adapter. This time interval allows sending repair requests in batch mode to the service adapter. Such a request is sent to re-activate a DFI collection after a collection error.
ServiceOperationsTimeout	%%SCServiceOperationsTimeout(ms)%%. (Global variable) 60000ms	Operation timeout (in milliseconds) for all service requests towards the remote Service Adapter.
SAProvidesAtLeast1DFMeasureAfterTimeout	%%SCSAProvidesAtLeast1DFMeasureAfterTimeout%%. (Global variable) 50000ms	Service Adapter timeout (in milliseconds) to provide at least one Data Feeder measure.
RetrieveDFMeasuresServiceRetryInterval	%%SCRetrieveDFMeasuresServiceRetryInterval%%. (Global variable) 10000ms	Retrieve Data Feeder measures service retry interval (in milliseconds).
ControlDFBundlesMaxSize	%%SCControlDFBundlesMaxSize%%. (Global variable) 150	Maximum control (start, stop or repair) Data Feeders per service operation.
RetrieveDFMeasuresBundlesMaxSize	%%SCRetrieveDFMeasuresBundlesMaxSize%%. (Global variable) 60000ms	Maximum size of Data Feeder Measures bundles retrieved from the Service Adapter.

ShouldLogDFMeasuresIncompatibleWithState	%%SCShouldLogDFMeasuresIncompatibleWithState%%. (Global variable) False	The flag which indicated if the Proxy should log the Data Feeder measures, which are incompatible with the associated Data Feeder collections' current states on the Proxy.

/tibco/private/adapter/ServiceCenter/ServiceAdapters/ServiceAdapters/SaProxy/v1_2/<**applicationname**>_config/DataFeederDefConfigList/InternalReference/<**Data Feeder Def**>



```
extendedProperties
    pollingPeriod = "%%SCDataAcquisitionPeriod(min)%%"
    XMLValidationMode = "%%SCXmlMsgValidation%%"
    RequestRepliesNbRetry = "%%SCInternalRequestRepliesRetryNb%%"
    internalRequestRepliesTimeout = "%%SCInternalRequestRepliesTimeout(ms)%%"
    noValueMsgWhenCollectionDown = "false"
    disableParamsWithInvalidSubscriber = "false"
    QuietMode = "False"
    ConnectorList
    DataFeederDefConfigList
        InternalReference
            OvisHTTP
                DfdVersion = "v1_0"
                NoValueWhenParameterIsNotAvailable = "True"
            Sql_MMediaDF
                DfdVersion = "v1_0"
                NoValueWhenParameterIsNotAvailable = "True"
            TeMIPFaultStats
                DfdVersion = "v1_0"
                NoValueWhenParameterIsNotAvailable = "True"
    CollectionThreadPool
    CollectionScheduler
    faultTolerance
    MonitoredConnectorList
    MonitoredDataFeederList
```

In the **DataFeederDefConfigList**, the following parameter can be configured:

Variable Name	**Default**	**Description**
NoValueWhenParameterIsNotAvailable	True	When “True”, this variable determines if a “no value” is returned when a parameter value has not been retrieved from the database. If “False”, the parameter is not encoded in the performance message.

2.5.2 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 SQM components

Dump

As for the other 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

2.6 Starting / Stopping SA proxy

Starting and stopping an Service Adapter proxy application is done through the standard 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 SQM environment variables

The commands are as follows:

- To start the application:

```
temip_sc_start_application -platform <platform name> -
director <director name> -application <application name>

   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.
```

- To stop the application:

```
Temip_sc_stop_application -platform <platform name> -
director <director name> -application <application name>
```

```
    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.
```

2.7 Deployment

2.7.1 Application distribution

As the SA proxy will need to be configured to connect to Web Container applications, run on a different system from the SQM SLM Primary Server, support multiple DFIs on multiple applications, it is really important to plan in advance, where it will be installed and how it will be configured to provide the best performance for data acquisition.

As described in the *SQM Administration Guide*, a SQM platform configuration can be distributed on several hosts. Applications can be logically grouped into platform directors.

Several SA proxy applications can run on the same host.

Several versions of the same SA proxy can also run on the same host.

Even if there is no restriction concerning the installation of SA proxy s on a system, you can group SA proxy applications into SQM directors using the following criteria:

- Technology driven. You use one director for each SA proxy, meaning that all the applications of the same SA proxy share the same director.
- OS driven. You use one director for each OS, so that, for example, all the SA proxy applications on Windows belong to Windows director and all the SA proxy applications installed on HP-UX system belong to the HP-UX system director.
- Geography driven. You use one director (or host) for each location, so that, for example, all SA proxy applications collecting on databases located in Paris belong to the “Paris” director.

To group SA proxy applications into directors, keep in mind that all applications of one director can be started or stopped in one command on that director, you should group your applications in the same director considering that each time the database is restarted they can all be restarted at once.

2.7.2 Load balancing

Even if the number of applications is not limited on a SQM host, and the number of DFIs supported by a SA proxy can be important, to optimize performances and to be able to support the collection load, the following configuration points have to be considered:

- Number of DFIs supported by a SA proxy application
- Number of applications running on a single system
- The performance of the targeted remote Service Adapter

To have the best possible configuration, the following parameters can be tuned:

- The Number of DFIs per SA proxy Application can be defined at the DFI discovery and filtering phase. The user may group the DFIs of a SA proxy application:

  a. Per DFD

  b. Per DFI property (MRP)

  c. Per customer

Note

At DFI load balancing configuration, make sure that the ratio of DFIs per SA proxy application is correctly balanced (avoid having an oversized application compared to other SA proxy applications on the same system)

- Depending on the system sizing (CPU, Memory…), the number of SA proxy application running on a single system has to be tuned. Please refer to the ***SQM Planning Guide*** document for more information.

Refer to the **Advanced application configuration** chapter for more information about these variables.

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="slmv11_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="slmv11_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="slmv11_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 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_2/`**`<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_2/<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_2/`***`<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

Proxy Service Adapter trouble shooting

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_2/config/<platform>_<director>_<application>.properties

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

Discovery tool trouble shooting

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_2_<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_2/<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
DFI	Data feeder instance
DF	Data feeder = Data feeder instance
MRP	Measurement reference point
SAI	Service Adapter Application Name (or Service Adapter instance)
SLA	Service level agreement
SLM	Service level management
SLO	Service level objective
SRM	Service Repository Manager
XML	eXtensible Mark-up Language