# HP OpenView
# Service Quality Manager


## SQL Service Adapter Toolkit

## Installation, Configuration and User's Guide

**Edition: 1.4**

**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 (SQM) SQL SA Toolkit. The Toolkit provides a user-friendly tool to customize an SQL Service Adapter. The SQL Service Adapter, once customized, is able to connect to an SQL database and collect raw data used by SQM to measure the quality of service.

This document describes how to:

- Install the SQL SA Toolkit
- Customize an SQL Service Adapter using the Toolkit
- Generate the SQL Service Adapter customization
- Deploy the SQL Service Adapter customization

## 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
- SQL database administration and operations
- Service Level Management

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

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

## Software Versions

The software versions referred to in this document are specified in “SQL Service Adapter Toolkit Installation”, section 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://www.hp.com/managementsoftware/support

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

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://www.managementsoftware.hp.com/passport-registration.html.

# Chapter 1

# Introduction

The hp OpenView Service Quality Manager (SQM) uses the SQL Service Adapter to collect Quality of Service (QoS) indicators data from SQL databases and expose them to SQM. The SQL Service Adapter is a generic Service Adapter. After a customization and configuration process, this Service Adapter is able to:

- Access an SQL database, tables, and columns,
- Collect data from the database,
- Map data to user defined Data Feeder parameters,
- Publish the collected data into SQM.

This section provides a brief overview of SQM and its acquisition layer, describes the SQL Service Adapter functionalities and the role of the SQL SA Toolkit.

For a detailed description of SQM, refer to the *hp OpenView Service Quality Manager Overview*.

For a detailed description of Service Adapters, refer to the hp OpenView Service Quality Manager Service Adapter User's Guide.

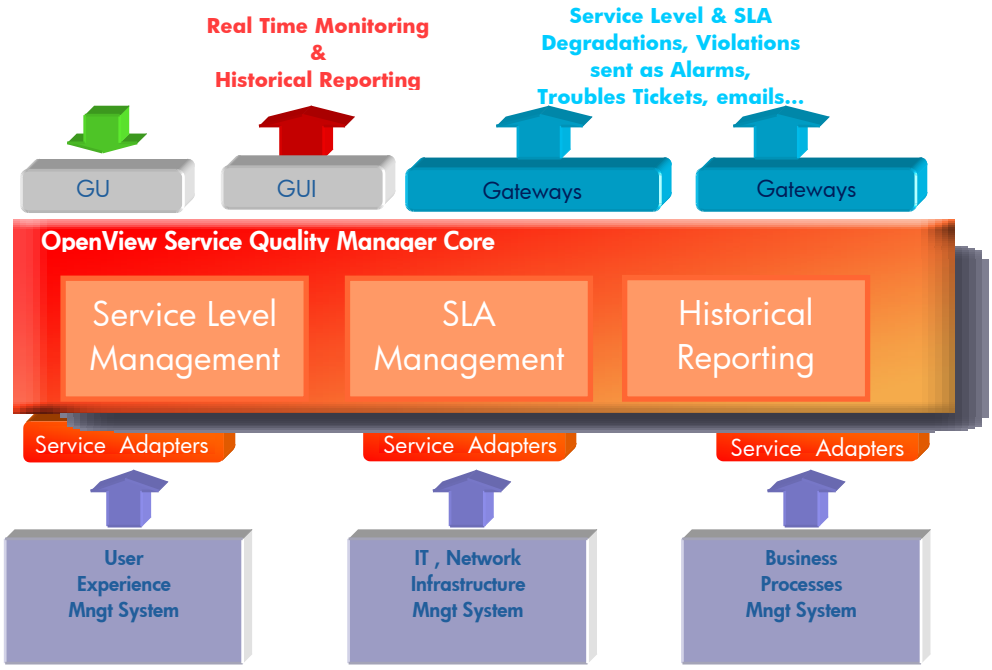
## 1.1 Service Quality Manager

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

SQM monitors the service quality by aggregating information coming from all data sources, such as the network, 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).

Figure 1 shows the main SQM components:

**Figure 1 Service Quality Manager Main Components**



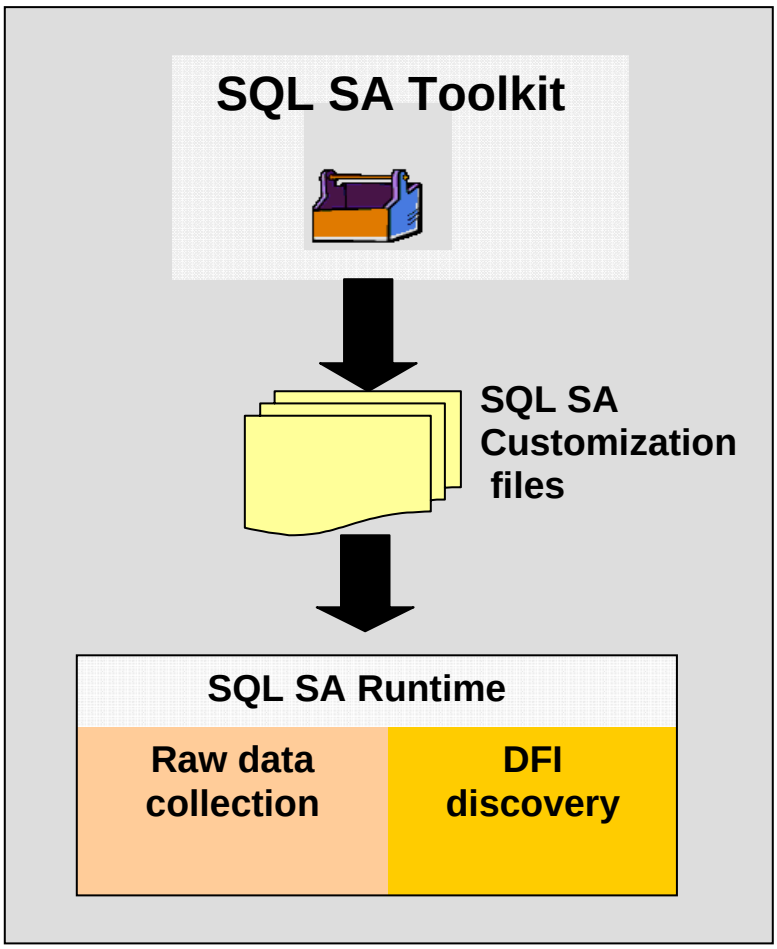

Service Adapters interface with SQM Core applications to:

- Retrieve configuration information and definitions (for instance, how to connect to an SQL database as well as the way to map the raw data from SQL to the SQM parameter format),
- Publish the collected QoS indicators used by the Service Level Management, SLA Management and Historical Reporting layers.

The SQM Core layers receive and store all measurement data from the Service Adapters, then calculate the status of each service to determine whether a service has failed to meet a service level (SL).

## 1.2 SQL SA and SQL SA Toolkit product components

This following picture presents the different components of the SQL SA and the role of the SQL SA Toolkit.




- The SQL SA Toolkit provides a tool to customize an SQL Service Adapter. The customization consists in defining the configuration parameters that will be used by the SQL Service Adapter to connect to an SQL database and collect raw data.
- The SQL Service Adapter component provides generic data collection and discovery functions:

The SQL SA Collection function performs a SQL query to the targeted database to collect the parameters.

The SQL SA DFI discovery identifies the points of measure on which the values will be collected. Once discovered, the definition of these points of measure is loaded into SQM.

### 1.2.1 SQL Service Adapter

The SQL Service Adapter acts as a bridge between SQM and the SQL database:

collecting QoS data from SQL tables,

exposing the SQL data values through Data Feeder Definition parameters,

Feeding SQM with the collected indicators.

#### 1.2.1.1 Definitions

The following chapters summarize the main definitions associated to Service Adapters and applied to the SQL Service Adapter. For more information about these important SQM notions, please refer to the *hp OpenView Service Adapters User's Guide*.

##### Data Feeder

A Data Feeder identifies a specific measurement point which provides a set of indicators (called parameters) about the quality aspect of a resource type (for example, Response Time from a given service access point).

The SQL Service Adapter can expose one or several Data Feeders. Each Data Feeder is associated to a SQL database table.

##### Data Feeder Definition

The Data Feeder Definition (DFD), for an SQL Service Adapter, contains the logical description of the parameters that are fed by an SQL table.

The following information is mandatory for each DFD:

- A name
- A version
- A set of data feeder definition parameters

A parameter can be considered as global (not linked to any subscriber information) or customer specific (collected for a particular customer). When a parameter is customer specific, one parameter can have different values for different customers.

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

If a parameter is customer specific, another column of the same SQL table should provide the customer or subscriber information associated to the collected data.

- A set of properties.

Properties are not parameters (not performance indicators). They identify the context of the collected parameters. The value of the property is set at the time of configuration of the data feeder instance.

- A Measurement Reference Point (MRP) naming schema.

The MRP naming schema is the way to build a unique identifier (based on concatenation of DFD properties) of each point of measure.

##### Data Feeder Instance

A data feeder instance (DFI) publishes the collected values according to a data feeder definition for one specific measurement point.

When a DFD parameter is customer dependant, the DFI collects the perceived QoS from a MRP for any subscriber of a specific customer.

Data Feeder Instances are created as follows:

- They are pre-registered by Service Operators when service adapters are not yet deployed (pre-registration is done through the SL Administration User Interface)
- They are discovered automatically by the Service Adapter and declared to SQM.

The SQL Service Adapter offers a DFI auto discovery feature. If requested, it is able to connect to the SQL database, collect each DFI property value(s) and declare each discovered DFI to SQM.

The SQL Service Adapter discovery feature also offers the capability:

- To filter the discovered DFIs (select only some of the discovered DFIs to be declared to SQM and supported by the SQL Service Adapter application),
- To synchronize the discovered DFI with SQM repository.

### Timestamp Management

There are two cases:

- The timestamp is present as a column in the SQL table. This information is used directly by the SQL Service Adapter for publishing collected values and identifies the values last inserted into the database (difference between the timestamp and last collected timestamp).
- there is no measurement time indication in the SQL table and in such case the SQL Service Adapter will use its polling current time in order to timestamp the collected data.

For a standard collection of the SQL Service Adapter, the timestamp information has to be present in the SQL database and provided at the SQL Service Adapter customization. If no timestamp is provided, a specific column containing this information can be created through an SQL view or script. Please refer to the **Advanced customization** chapter for more information.

### Customer and Subscriber

As previously mentioned, DFD parameter values can be collected for a specific customer (organization or corporation) or for a specific member of this organization (a user or a subscriber). The SQL Service Adapter is able to retrieve QoS data directly for a customer, or for a subscriber.

The customer or subscriber information has to be a column of the SQL table associated to the DFD.

To make the mapping between subscribers and customers, OpenView SQM uses subscriber IDs. The subscriber ID identifies the user of the service in a service quality measure. The subscriber ID can be an IP address, a user login name, or an identifier directly linked with a user, such as a DSL port number.

Sometimes, the user happens to be directly the Customer.  So, the subscriber ID maps directly to the customer ID. In such case, that does not require a mapping, the domain is by default the generic “ServiceCenter” domain.

To associate a Subscriber ID to a customer, a mapping structure is necessary (available in the SQM Central Repository). The structure contains:

- The customer name
- The naming plan. A naming plan defines the customer identifiers in each sub-domain. A domain logically groups customer identifiers.

Here is an example of mapping:

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

Refer to the ***SQM Information Modeling Reference Guide*** for more information about Subscriber ID mappings.

**Note**

For customer or subscriber dependant parameters, one column of the DFD table has to contain the customer or subscriber information.

If a subscriber column exists, it has to provide subscriber IDs for a single subscriber domain.

### Connection definition

The connection is a configuration component of the SQL Service Adapter. This component contains all necessary information for the SQL SA to connect to the SQL database.

The following parameters are requested to define the SQL SA connector:

- ***JDBC driver***: should be the driver that has been installed to connect to the database
- ***Database URL***: A JDBC URL provides a way of identifying a data source so that the appropriate driver will recognize it and establish a connection with it.
- ***User Identifier***: login name to access the database
- ***User Password***: associated login password.

The connection information is provided at the configuration of the SQL Service Adapter. The user identified for the connection should have enough access rights to extract the data from the database table or view.

### Example

The following chapter will illustrate the previous definitions with a concrete use case.

For instance, a Performance Agent is installed on several systems to collect the following performance indicators and store them into an SQL database:

- CPU usage
- Disk Usage
- Memory usage

Each indicator is collected per user and per system.

In our example, the SQL Service Adapter will access an SQL table to collect statistics on system resource usage.

The collected parameters are CPU Usage, Memory usage and Disk Usage. Each parameter is represented as a table column in the database.

To uniquely identify from where data are collected, the "***System Name***" property is used as the MRP.

As the parameters are collected per user connected on the system, the user (subscriber) information is present in the SQL table.

The SQL table associated to the previously defined Data Feeder will be created as follows. If this is not the case, some pre-customization steps (such as view creation) will have to be performed

| | DFD Parameters | | | MRP property | Subscriber | DFD Properties |
|---|---|---|---|---|---|---|
| **Timestamp** | **CPU_USE** | **MEM_USE** | **DISK_USE** | **SYSTEMNAME** | **USER** | **IPADDRESS** |
| 2003-10-23 10:00:00 | 30.00 | 500 | 5000 | host1.emea.hp.net | grant | 15.23.456.34 |
| 2003-10-23 10:05:00 | 23.00 | 450 | 5000 | host1.emea.hp.net | smith | 15.23.456.34 |
| 2003-10-23 10:00:00 | 50.00 | 395 | 4598 | host2.emea.hp.net | grant | 16.156.67.15 |
| 2003-10-23 10:05:00 | 20.00 | 630 | 6879 | host2.emea.hp.net | smith | 16.156.67.15 |

In the table, two MRP property values which identify two Data Feeder Instances are currently available: **host1.emea.hp.net** and **host2.emea.hp.net**.

This use case will be used to illustrate the following SQL SA Toolkit customization steps.

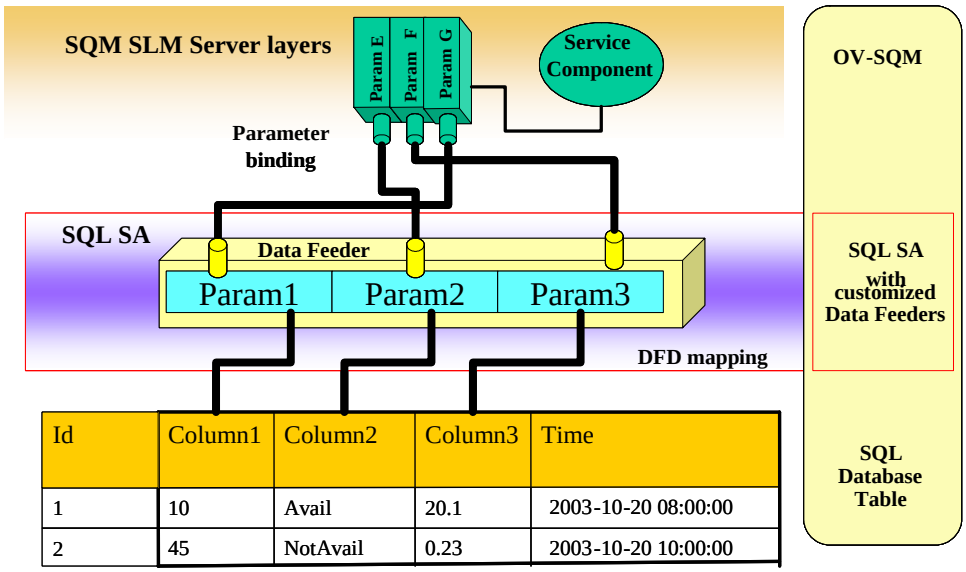
### 1.2.1.2 Data collection overview

The SQL Service Adapter connects to the database through the JDBC driver provided by the database manufacturer. This driver implements the standard JDBC API that is used by the application to send the SQL queries and collect the raw data. The SQL Service Adapter is also in charge of opening and managing the physical connections with the database, especially in case of unexpected disconnection.

Depending on the physical configuration and the environment requirements, the Service Adapter will need to be installed on the SQL database server, or need to access the database remotely. Both configurations can be considered, but some restrictions need to be checked:

- If remote access is possible, the connection to the dedicated database port has to be validated.
- If local access is mandatory, the SQL-Service-Adapter-supported Operating System must be checked.

The SQL Service Adapter uses a polling mechanism to collect the performance data. At each polling period, the Service Adapter connects to the database to perform an SQL collection query that will retrieve data values. The collected data are then mapped to the defined DFD parameters and published with timestamps to SQM Core layer (Service Level Monitoring). The DFD mapping definition is one of the customization steps of an SQL SA customization that is illustrated by the following picture.




### 1.2.1.3 DFI discovery overview

The SQL Service Adapter provides the Data Feeder Instance discovery capability. This functionality consists in retrieving from the SQL database the list of existing rows related to Data Feeder Instances. It uses a SQL discovery query. This list of DFIs can be filtered by the user (to support only a subset of instances) and loaded into the SQM Service Repository Manager. Once declared in the repository, the Data Feeder Instances can be linked to SQM Service Component Instances (binding) using the SQM SL Administration User Interface, and the collection can be activated.

### 1.2.1.4 Database requirements

#### Supported operating systems

The SQL Service Adapter v1.3 is running on the following Operating Systems:

- HP-UX 11.11
- Windows XP

#### Supported SQL databases

The SQL Service Adapter V1.3 supports the following SQL databases:

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

If the support of other databases is necessary, the user may request the validation of the new database to their hp Sales Representative. For new database support, the minimum required is that the SQL database is accessible through a JDBC driver version 3.0.

### Supported SQL data types

The following table associates a supported SQL data type to each SQM data type:

| SQM data type | Database data type | | | |
|---|---|---|---|---|
| | Oracle | Sybase | SqlServer | MySQL |
| String | VARCHAR2 | varchar<br>smalldatetime | VARCHAR | DATE<br>VARCHAR |
| Enum | NUMBER<br>INT | Int | INT | INT |
| Float | FLOAT | Float | FLOAT | FLOAT |
| Int | NUMBER<br>INT | Int<br>smallint | INT<br>smallint | INT |
| AbsTime | TIMESTAMP | datetime | datetime | TIMESTAMP |
| RelativeTime | INT,INTEGER | Int | INT | INT |

This list of SQL supported data types can augment. Please contact the ***SQM Support Team*** to have an updated list of supported databases and data types.

Any timestamp column that has to be collected by SQL Service Adapter must be previously formatted to a GMT time.

**Example**

For example, the Timestamp column for an Oracle database, has to be defined with **TIMESTAMP** data type, and its value can be calculated with the following Oracle function: **SYS_EXTRACT_UTC(SYSTIMESTAMP)**

If a parameter is collected from a table column whose data type is not supported, a specific view or script has to be created to convert the column data type to a supported data type (refer to the **Advanced customization** chapter for more information about view creation). If there is no possibility to convert the column data type to a supported type, a specific function has to be developed in the SQL Service Adapter to support the specific type. For such advanced development request, please contact your hp Sales Representative.

**Note concerning SQM RelativeTime data type**

By default, a SQL column maps into a SQM RelativeTime data type must be expressed in second in the SQL database.

Anyway the SQL SA can be configured to consider that the SQL column represent a relative time expressed in milliseconds. To configure the SQL SA, edit its property file located in:

#### On Windows:

```
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\properties\TeSC<SACustomName>.properties
```

#### On Unix:

```
$TEMIP_SC_HOME\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\properties\TeSC<SACustomName>.properties
```

Add the following property to the file:

parameter.relativetime.insecond=false

Setting this property all relative time parameters are considered as being expressed in milliseconds in the SQL table.

### 1.2.2 SQL Service Adapter customization

In order to dramatically shorten the time necessary to customize an SQL Service Adapter and allow easy modifications of the SQL Service Adapter collection, the SQL SA Toolkit provides a powerful and user-friendly customization environment. The customization of an SQL Service Adapter does not require any coding, only definition and integration tasks.

The SQL SA Toolkit comes with a Graphical User interface to go through the following SQL Service Adapter customization steps:

- Define SQL SA customization project

The definition of an SQL SA customization project is the primary step of the customization with the toolkit. The project is a file that contains all the SQL SA Customization properties (SA description, SQL connection definition, DFD mappings…).

- Configure SQL Service Adapter connector

Configuring the database connection consists in providing connection parameters necessary to access the SQL database.

- Model SQL Service Adapter Data Feeders and define collection and discovery queries

This modeling phase is composed of several subtasks:

*DFD Parameter definition*, which consists in providing the DFD name, id, version, description and associated SQL table.

*Data Feeder Parameter Mapping*, which consists in browsing the database table/view associated the Data Feeder and defining the mapping between the DFD parameter and the SQL table/view column.

*Data Feeder Property Mapping* consists in browsing the database table/view and mapping the table columns to DFD properties.

*The SQL Collection query definition* consists in declaring the SQL request that is performed to collect the parameter values from the database.

*The SQL Discovery query definition* is the declaration of the SQL request that discovers the Data Feeder Instances.

- 4. Generate SQL Service Adapter customization

This phase creates a customization zip file that is used to deploy the SQL Service Adapter customization.

Prior to these steps, it is necessary for the user to correctly identify the context, environment and requirements associated with this SQL Service Adapter customization. Chapter 3 of this document provides the full process flow of an SQL Service Adapter customization.

# Chapter 2

# SQL SA Toolkit installation

This chapter describes how to install the SQL SA Toolkit on HP-UX or Windows. After installation steps have been completed, please follow the instructions in Chapter 3 to use the SQL SA toolkit and customize an SQL Service Adapter.

## 2.1 Software Requirements

Operating System

- HP-UX 11.11 or Windows XP
- SQM Software
- HP OpenView Service Quality Manager V1.3 (Kernel subset)
- HP OpenView SA Common V1.3 (**SQMSAGTWCOMMON**)
- HP OpenView SQL Service Adapter Runtime V1.3 (**SQMSASQL**)
- JDBC drivers for the targeted SQL database
- The following URLs are pointers to the supported SQL Service Adapter JDBC drivers:

Microsoft SQL Server 2000 (msbase.jar, mssqlserver.jar, msutil.jar)

- http://www.microsoft.com/downloads/details.aspx?FamilyID=4f8f2f01-1ed7-4c4d-8f7b-3d47969e66ae&DisplayLang=en

Sybase

http://www.sybase.com/products/middleware/jconnectforjdbc

Oracle 8.1.7 (classes12.zip)

- http://otn.oracle.com/software/tech/java/sqlj_jdbc

Oracle 9.2.0.6 (Ojdbc14.jar)

- http://otn.oracle.com/software/tech/java/sqlj_jdbc

MySQL (mysql-connector-java-3.0.8-stable-bin.jar)

- http://www.mysql.com/products/connector-j/

## 2.2 Hardware Requirements

For hardware requirements, refer to the HP OpenView Service Quality Manager Installation Guide.

- Minimum of 6 MB disk space. 10 MB recommended.

## 2.3 Installing the Software

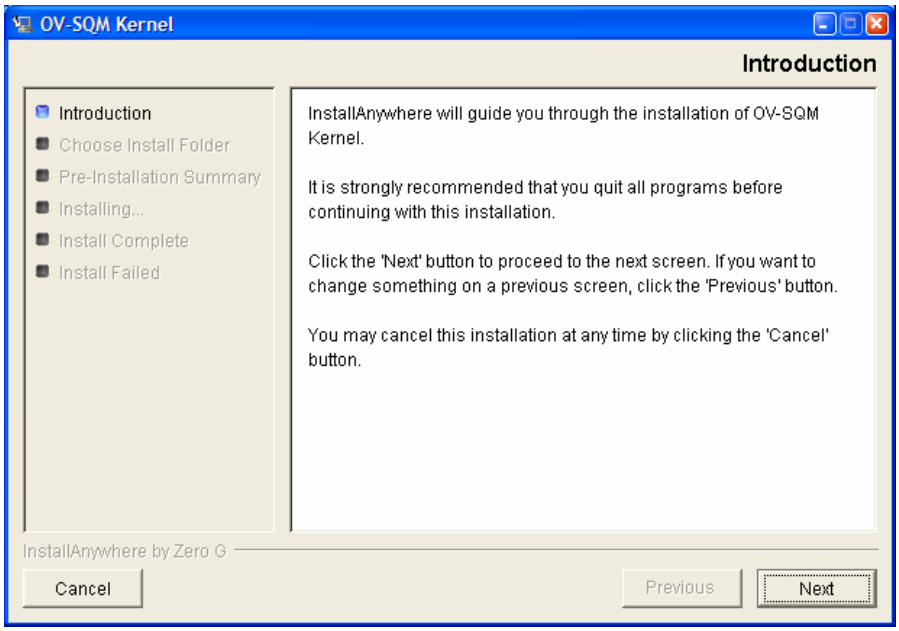
To install the SQL SA Toolkit, perform the following steps:

### 2.3.1 Installing on Windows

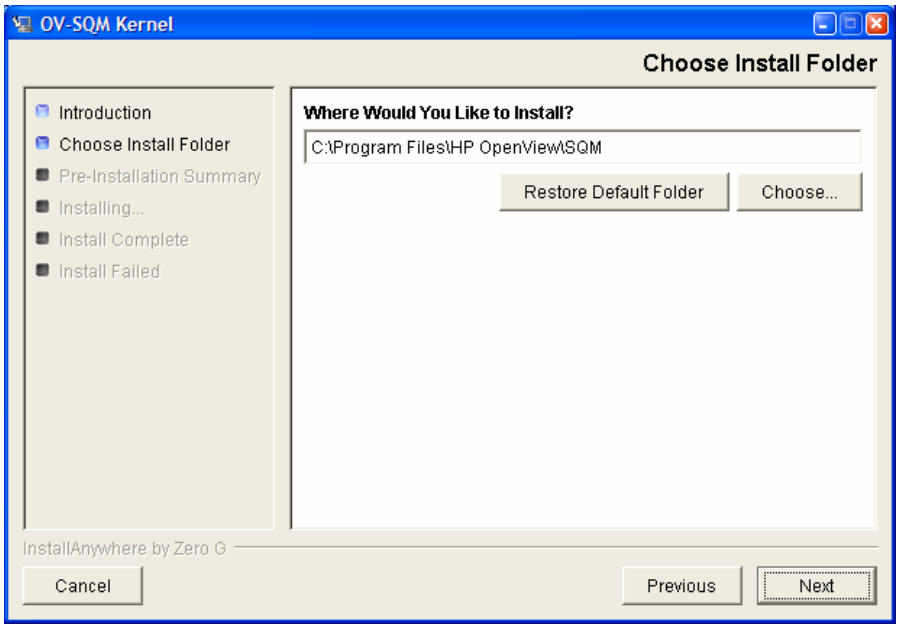
#### 2.3.1.1 Installing SQM Kernel Subset

The SQM Core Kernel subset is a pre-requisite for the installation of the SQL SA Toolkit.

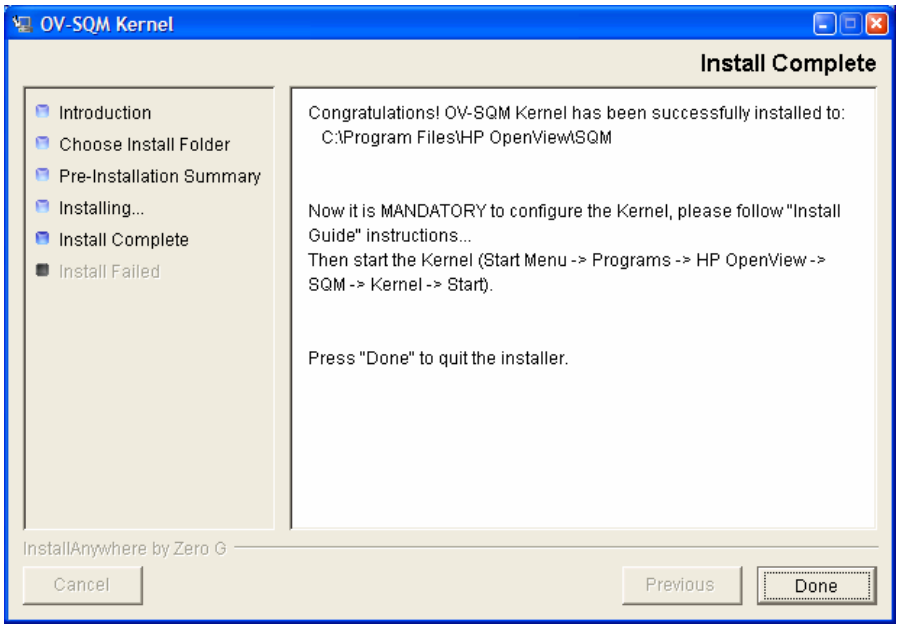
Insert the hp OpenView SQM Core CD in your CD-ROM drive, navigate in the `SQM-1.30.00-COREWIN\Windows\User_Interfaces` folder, and run the `SQMKERNEL-1.30.00.exe` installer.



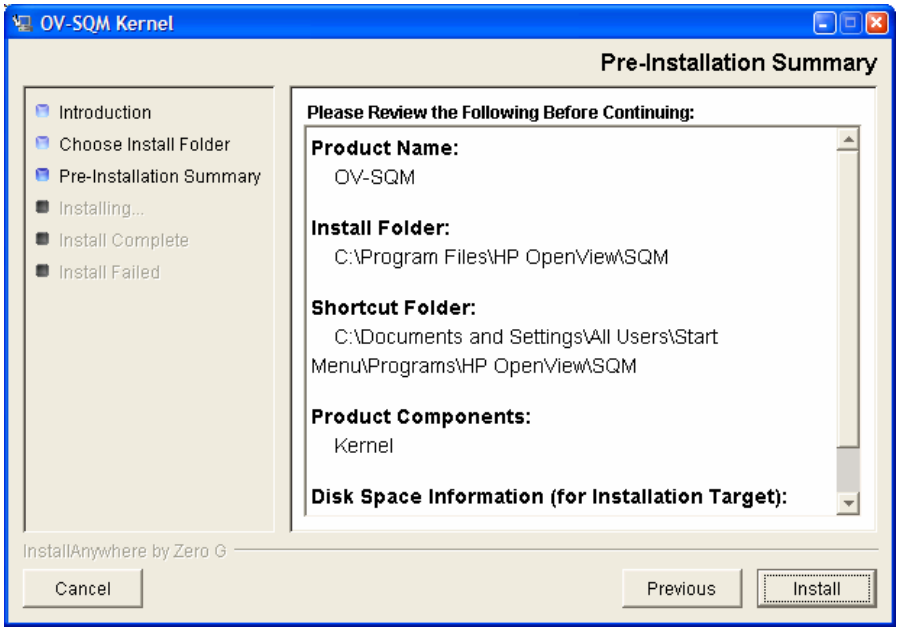

1. Click '**Next**' to proceed.



2. Select the destination directory of your OV SQM. If SQM has already been installed (for SLA Monitoring use for instance), you **MUST** install in the same directory. Click '**Next'** to proceed.




3. This window allows you to check the selected options. Click '**Install'** to perform else '**Previous'** to modify them.




4. Click '**Done'** to end installation process.

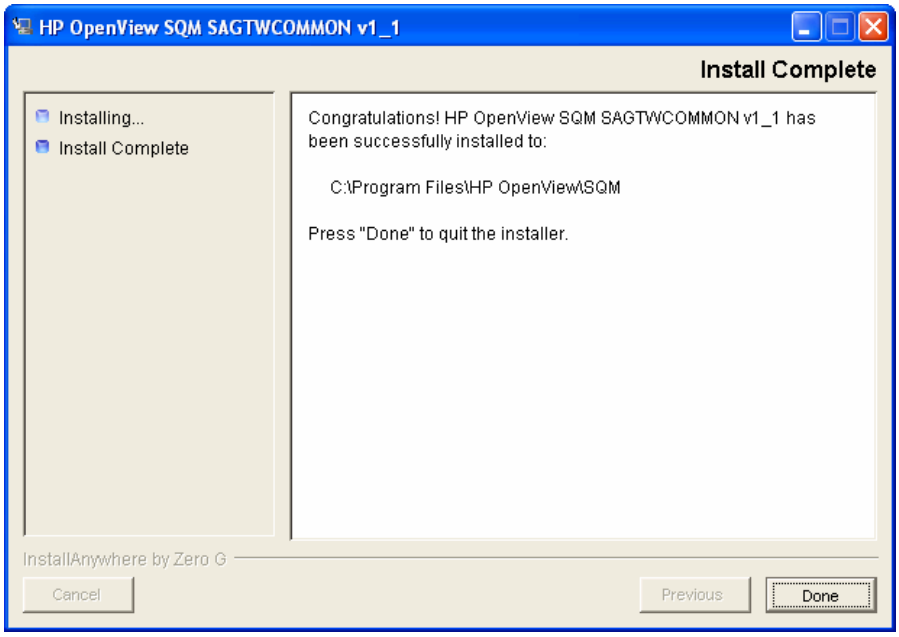
### 2.3.1.2 Installing the SQL SA Toolkit

To install the SQL SA Toolkit, perform the following steps:

First install the SA Common component if necessary (if already done go to **step 4**)

1. Insert the Service Adapters and Gateways media in your CD-ROM drive, navigate to the `SQM-1.30.00-SAGTW\Windows` folder, and run the `SQMSAGTWCOMMON-1.30.00.exe` installer.

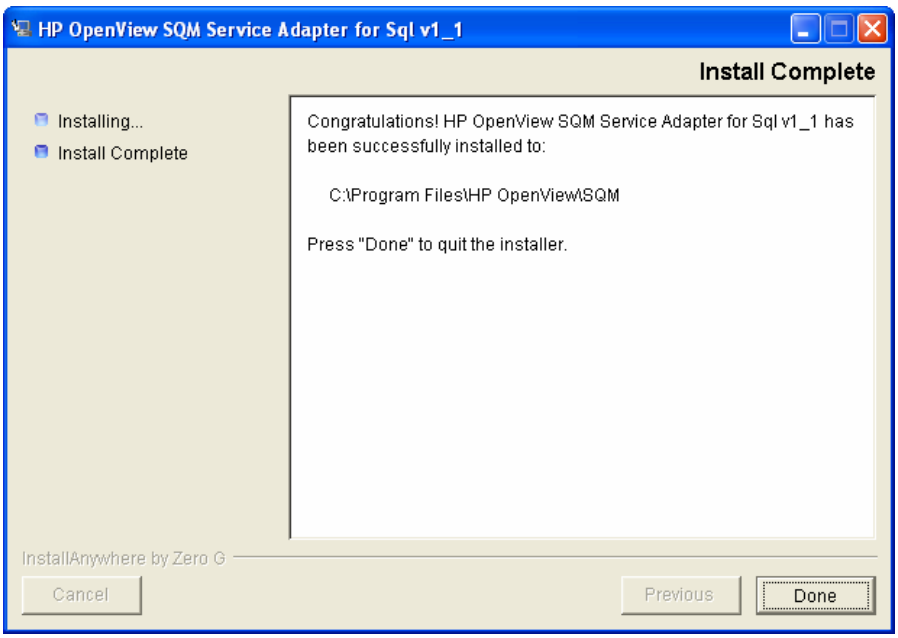
2. The software is installed and the **Install Complete** window is displayed.




3. To end the installation process, click **Done**

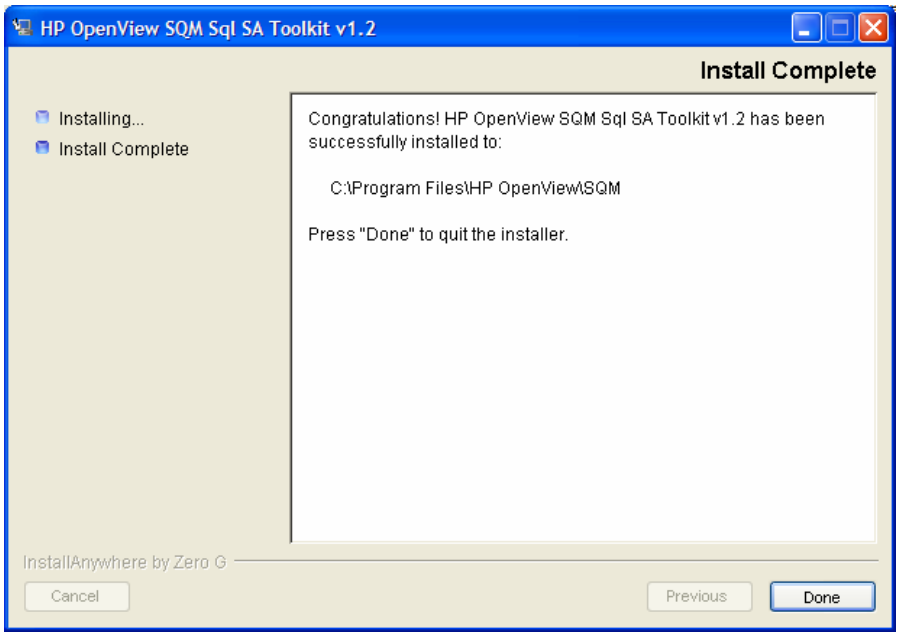
4. Run the `SQMSASQL-1.30.00.exe` installer. If this steps is already performed, switch to **step 7**)

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




6. To end the installation process, click **Done**.

7. Run the `SQMSQLSATK-1.30.00.exe` installer.
8. The software is installed and the **Install Complete** window is displayed.




9. To end the installation process, click **Done**.

## 2.3.2 Installing on HP-UX

### 2.3.2.1 Installing the SQM Kernel subset

The SQM Core Kernel subset is a pre-requisite for the installation of the SQL SA Toolkit.

1. Connect as “root” user.
2. Mount the hp OpenView SQM Core CD-ROM on */cdrom* :

```
# cd /cdrom
# sqm_install /opt/OV/SQM /cdrom/SQM-1.30.00/HPUX/KIT minimal
```

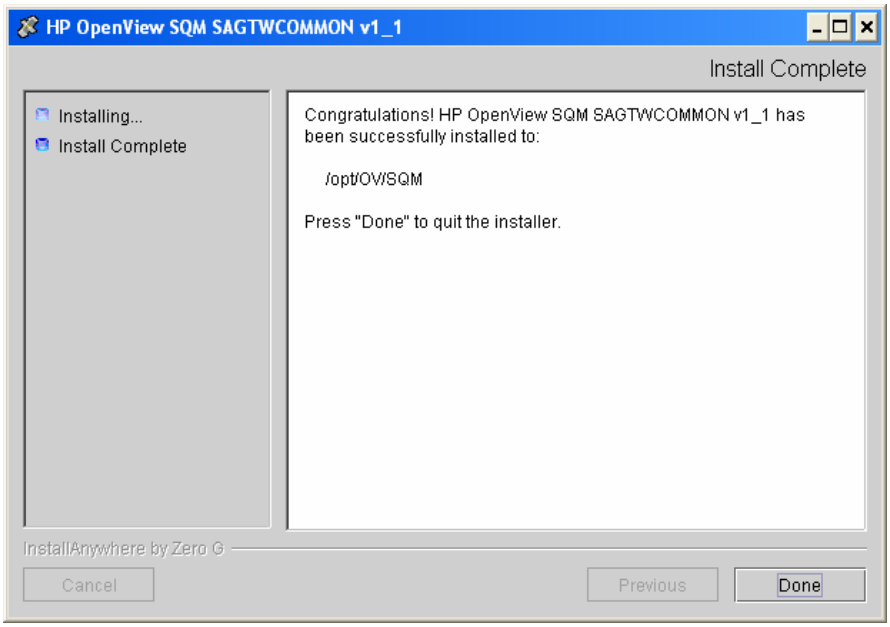
### 2.3.2.2 Installing the SQL SA Toolkit

To install the SQL SA Toolkit, perform the following steps.

3. First, connect as “**root”** user on the system.
4. Install the SA Common component if necessary (if already done go to **step 4**)
5. Mount the hp OpenView Service Adapters and Gateways CD-ROM on /cdrom directory and execute:

```
# cd /cdrom/SQM-1.30.00-SAGTW/HPUX
# export DISPLAY=<Your Display>
# export $TEMIP_SC_HOME=/opt/OV/SQM
# . $TEMIP_SC_HOME/jre/jre-setup.sh
# ./SQMSAGTWCOMMON-1.30.00.bin
```

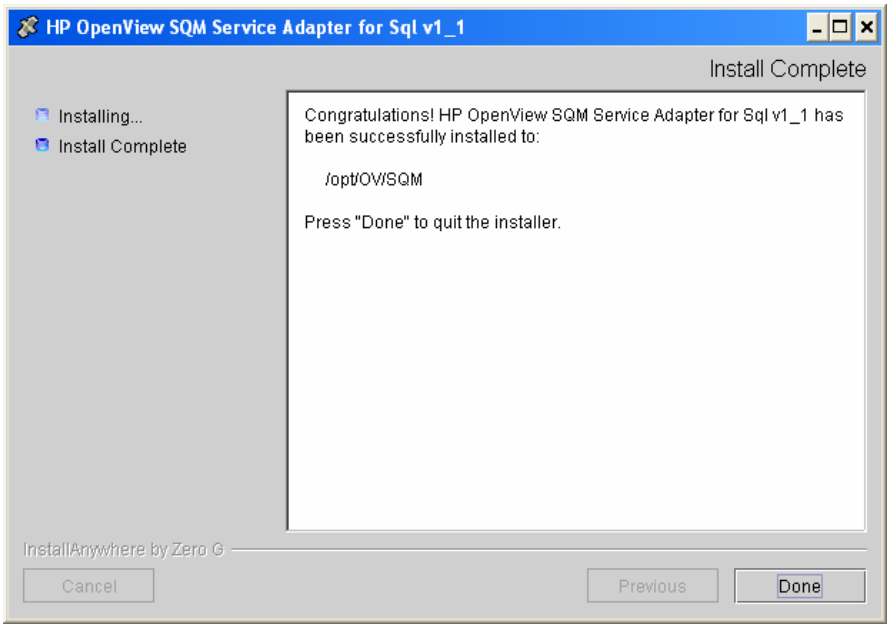
6. The software is installed and the **Install Complete** window is displayed.




7. Mount the hp OpenView SQM Service Adapters and Gateways CD-ROM on /cdrom directory and execute (If this steps is already performed, switch to **step 7**)

```
# cd /cdrom/SQM-1.30.00-SAGTW/HPUX
# export DISPLAY=<Your Display>
# export $TEMIP SC HOME=/opt/OV/SQM
# . $TEMIP_SC_HOME/jre/jre-setup.sh
# ./SQMSASQL-1.30.00.bin
```

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

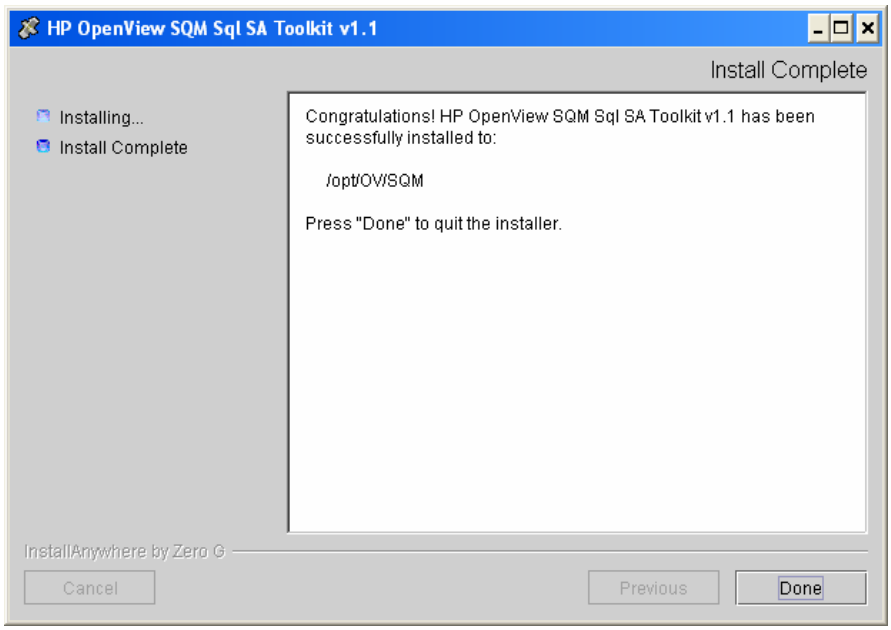



9. To end the installation process, click **Done**.

10. Mount the hp OpenView SQM Service Adapters and Gateways CD-ROM on /cdrom directory and execute:

```
# cd /cdrom/SQM-1.30.00-SAGTW/HPUX
# export DISPLAY=<Your Display>
# export $TEMIP SC HOME=/opt/OV/SQM
# . $TEMIP SC HOME/jre/jre-setup.sh
# ./SQMSQLSATK-1.30.00.bin
```

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



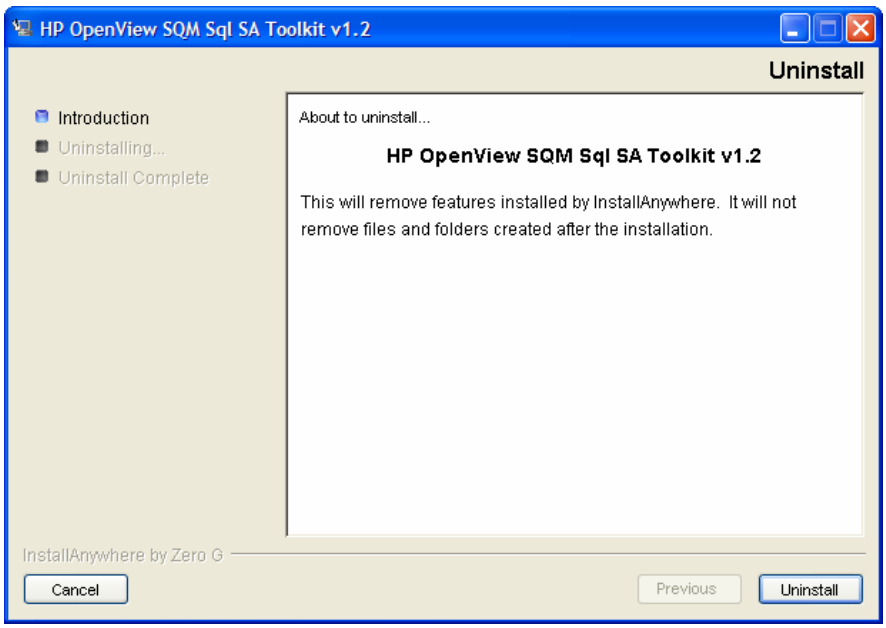

12. To end the installation process, click **Done**.

### 2.3.3 Uninstalling the Software on Windows

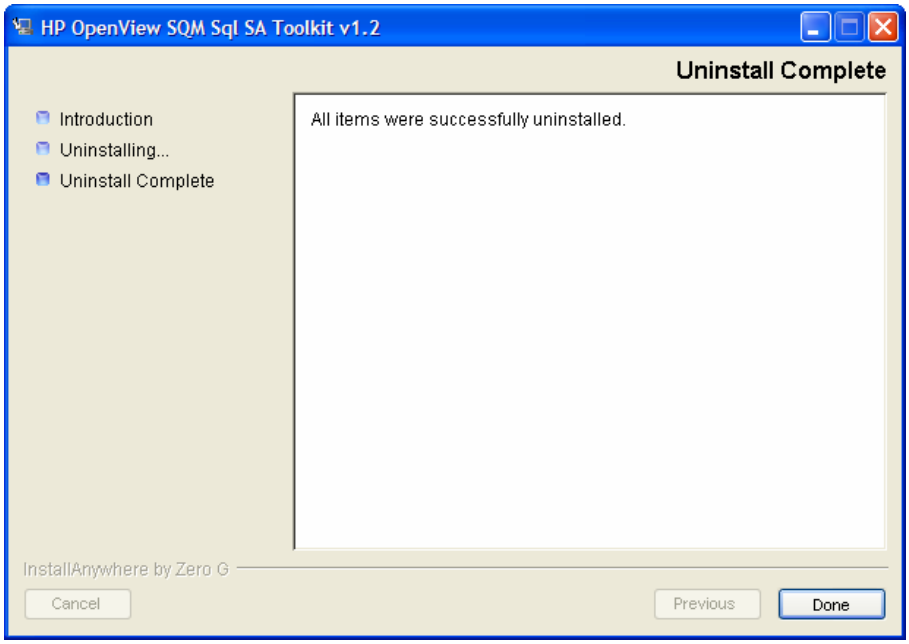
To uninstall the SQL SA Toolkit software:

1. Select menu: `All Programs -> HP OpenView -> SQM -> Sql SA Toolkit v1_2 -> Uninstall`

The **Uninstall** window is displayed.




2. Click **Uninstall**. The software is uninstalled from your system.

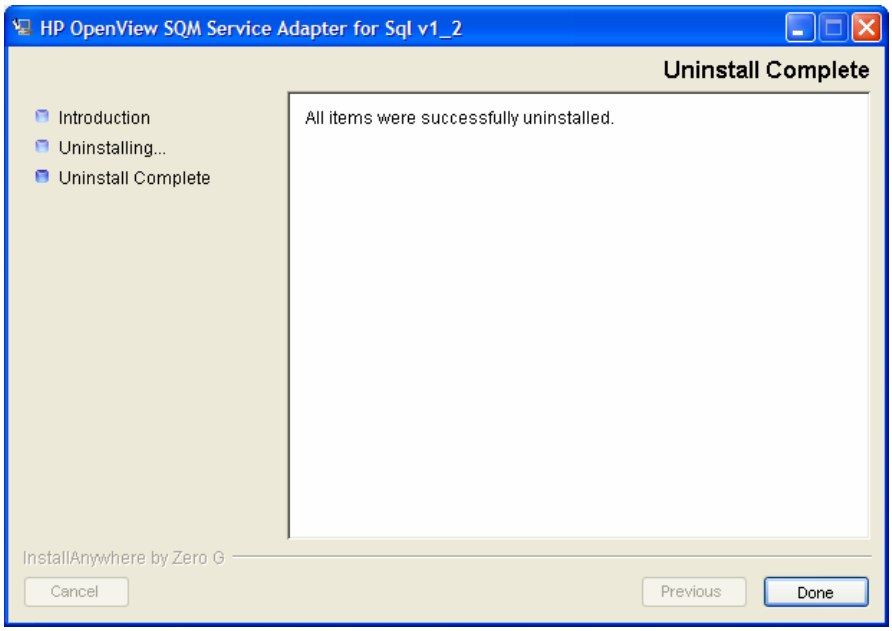



3. To finish, click **Done**.

   If no additional SQL Service Adapters are running on the system, un-install the SQL SA v1_2 runtime kit.

4. Select menu: `All Programs -> HP OpenView -> SQM -> ServiceAdapters -> Sql v1_2 -> Uninstall`

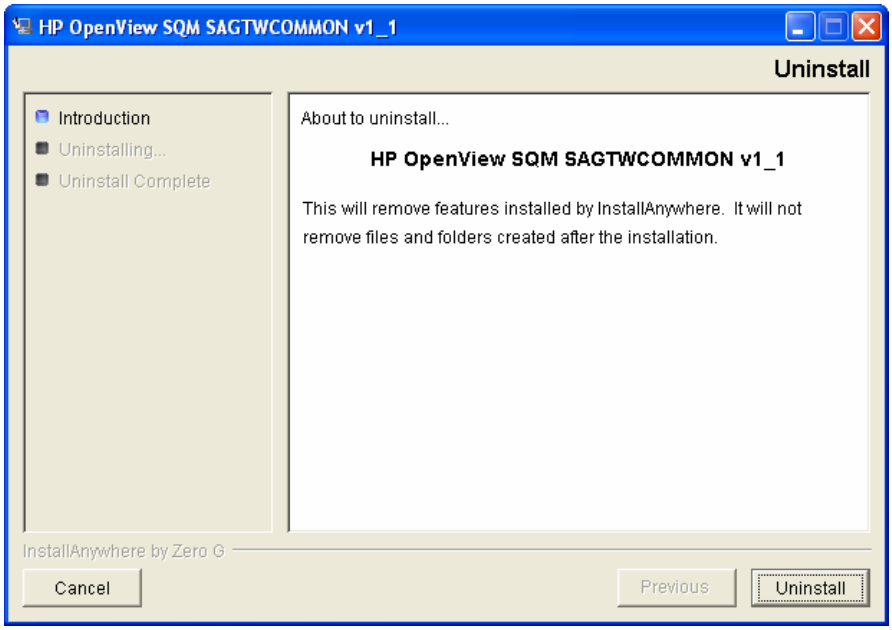

5. Click **Uninstall**. The software is uninstalled from your system.




6. To finish, click **Done**.

   If no Service Adapters or Gateways are running on the system, un-install the SA Common v1_2 kit.

7. Select menu: `All Programs -> HP OpenView -> SQM -> ServiceAdapters -> Common v1_2 -> Uninstall`



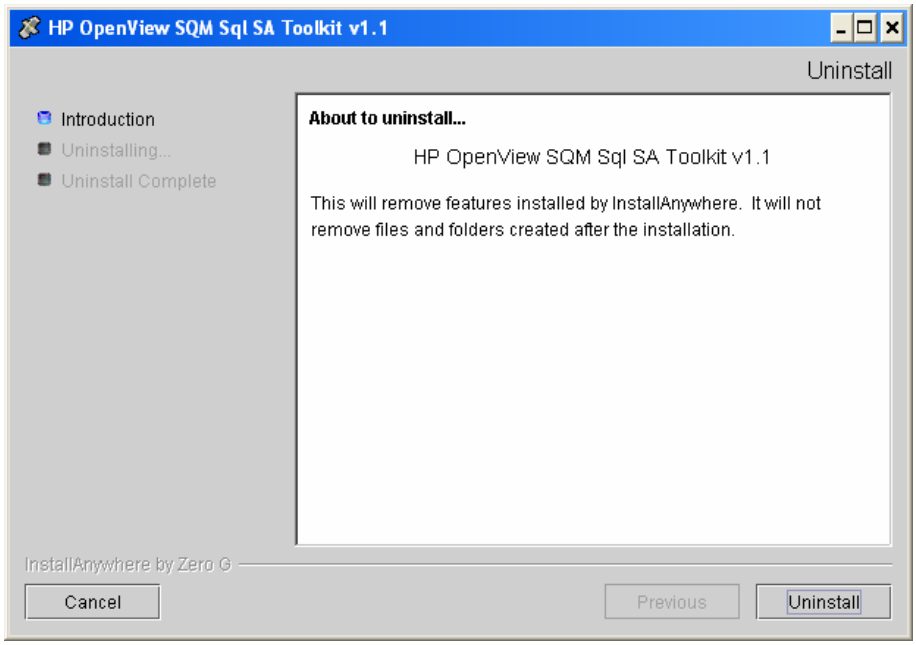

8. Click **Uninstall**. The software is uninstalled from your system.
9. To finish, click **Done**.

### 2.3.4 Uninstalling the Software on HP-UX

To uninstall the SQL SA Toolkit software:

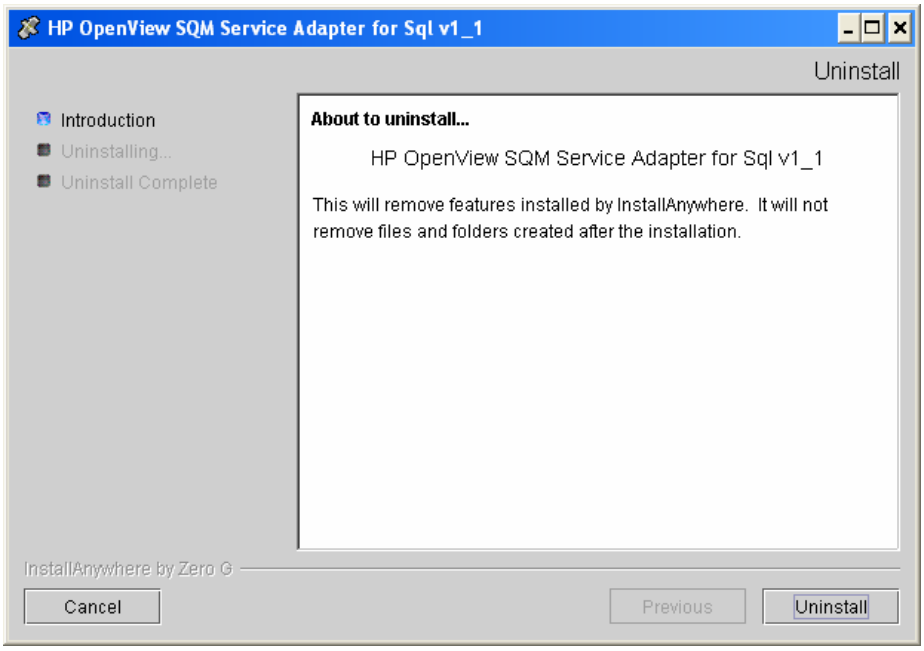
10. Log on as **root** user
11. Load the SQM environment variables (/var/opt/OV/SQM/slmv12/temip_sc_env.sh)
12. Perform the following commands on the SQM platform where the SQL SA Toolkit has been installed:

```
# cd $TEMIP_SC_HOME

# . /ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/Uninstall_SqlTk
```




13. Click **Uninstall**. The software is uninstalled from your system.
14. To finish, click **Done** in the following window.
15. If no additional SQL Service Adapters are running on the system, uninstall the SQL SA v1_2 runtime kit.

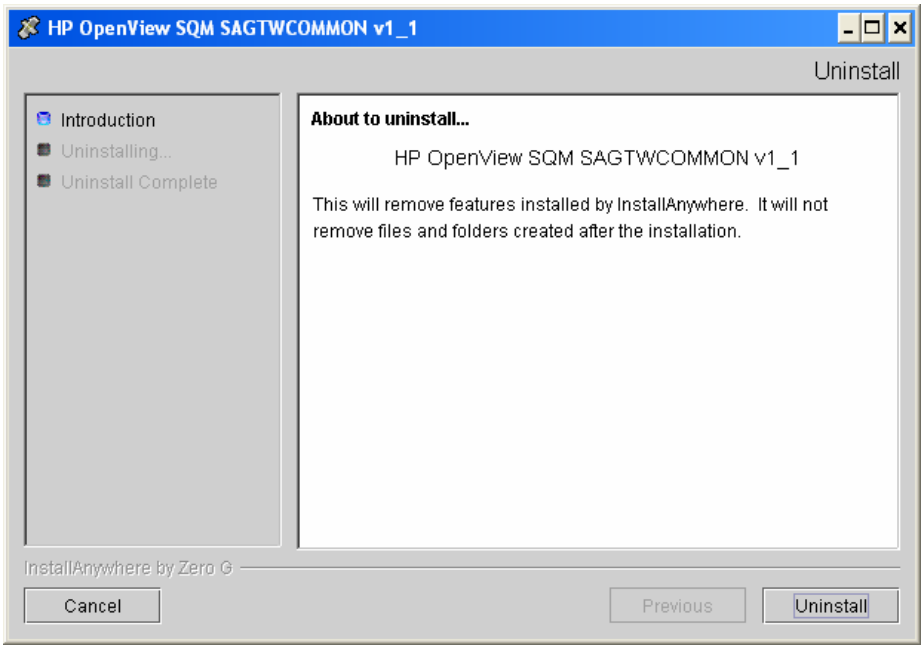
```
# cd $TEMIP_SC_HOME

# . /ServiceAdapters/Sql/v1_2/UninstallerDataSql /Uninstall_Sql
```




16. If no Service Adapters or Gateways are running on the system, uninstall the SA Common v1_2 kit.

```
# cd $TEMIP_SC_HOME

# . /ServiceAdapters/Common/v1_2/Uninstaller_SAGTWCOMMON
/Uninstall_SAGTWCOMMON
```

17. Click **Uninstall**. The software is uninstalled from your system.



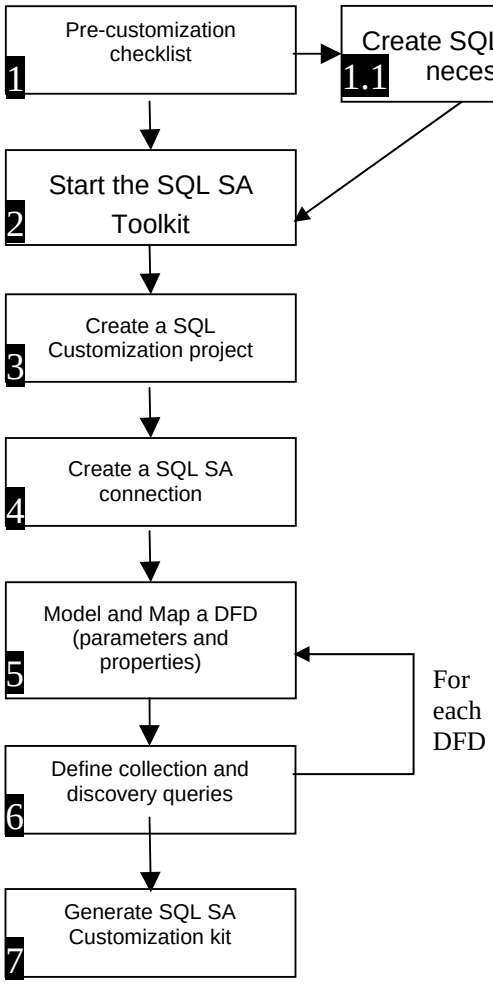

18. To finish, click **Done** in the following window.

# Chapter 3

# SQL SA Customization process

This chapter provides the mandatory steps to customize an SQL Service Adapter.

The following diagram summarizes the SQL SA Customization phases.



## 3.1 Validation check list, before customization project

Before starting the proper toolkit customization, it is necessary to specify the SQL Service Adapter, by:

- Fully understand the SQL SA development context
- Bringing out the expected environment requirements (targeted database type and version)
- Identifying the SQL database, tables and columns that will be used for collecting raw data (determine if a specific SQL view or script needs to be created to adapt to the SQL Service Adapter collection)

### 3.1.1 SQL database cookbook

The following questions need to be addressed before starting the customization.

| |
|---|
| What is the targeted SQL Database (ex: Oracle, Sybase…) and what is the associated version? |

The list of supported SQL databases and versions is provided in the chapter **Supported SQL databases**. A crosscheck is necessary to anticipate any support issue.

| |
|---|
| Can the Service Adapter be connected remotely to the SQL Database or does it need to be installed on the same host as the SQL Database?<br><br>–If the SQL Service Adapter has to be installed on the same system as the SQL database (due to access rights, firewall limitation), what is the targeted Operating System of the SQL Database? |

If the SQL Service Adapter has to run on the SQL database system, it is recommended to check the supported Operating systems in the chapter **Supported operating systems**.

| |
|---|
| What is the database schema (table/columns)?<br><br>Is there any information to be collected from several tables for one SQL Service Adapter DFD?<br><br>Is the data to be collected really available in the identified table columns (is there any specific calculation to perform on this data before publishing the parameter values in SQM)? |

The previous questions will help in validating the mapping between the SQM DFD parameters and the SQL table columns:

It is necessary to check that the expected DFD parameter values are really available in the SQL table columns. If any specific calculation is to be performed on the SQL Database raw data to retrieve the expected DFD parameter value, it is recommended to use SQL views or scripts to perform the calculation algorithm (see 5.2)

The mapping between the DFD and the SQL database table has to be straight forward, using the following rules:

- One DFD has to be associated to one SQL Table
- Each DFD parameter has to be associated to one Table column

- Each MPR property has to be associated to one Table column
- Several DFD properties can be part of the MRP
- If the database schema does not offer these mapping rules, it is necessary to consider the use of SQL views (see 5.2 section)

To perform the customization steps with the SQL SA Toolkit, it is recommended to retrieve a dump of the targeted SQL database to directly access the database information (table, view, columns) from the Toolkit User interface, and avoid any mapping mistakes.

## 3.2 Setting up the SQL SA Toolkit

From the previous chapter the targeted SQL database types and versions have been identified. The user will now need to retrieve the associated JDBC drivers that will be used by the Toolkit and the SQL Service Adapter to connect to the database.

**Note**

It is recommended to copy all the drivers on the system where has been installed the SQL SA Toolkit in the Directory:

**On Windows**

<SQM Installation directory>\ServiceAdaptersToolkit\Sql\v1_2\lib\jdbc

**On Unix**

<SQM Installation directory>/ServiceAdaptersToolkit/Sql/v1_2/lib/jdbc

Then the SQL SA Toolkit setup can be performed.

### On Windows

- `Select menu: All Programs -> HP OpenView -> SQM -> Sql SA Toolkit v1_2 -> Setup`

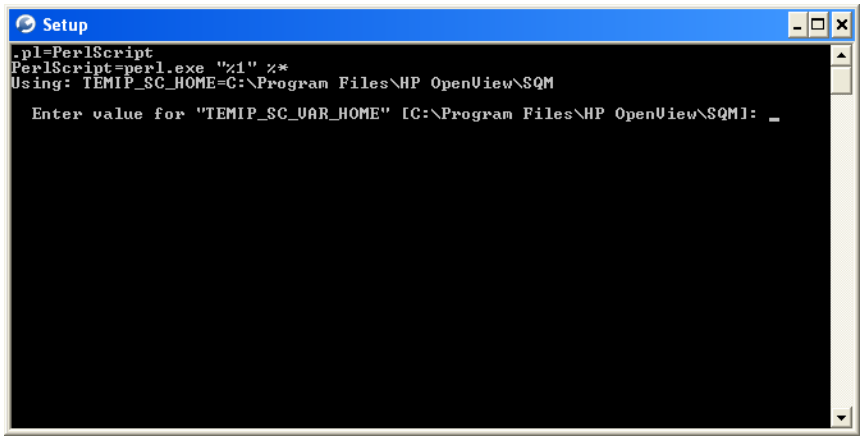
A Command Line window appears and prompts the user to set:

The TEMIP_SC_VAR_HOME environment variable if it is not defined. The corresponding directory is used by the SQL SA Toolkit for logging and tracing.

The SQM installation directory is proposed as a default value.

**Note**

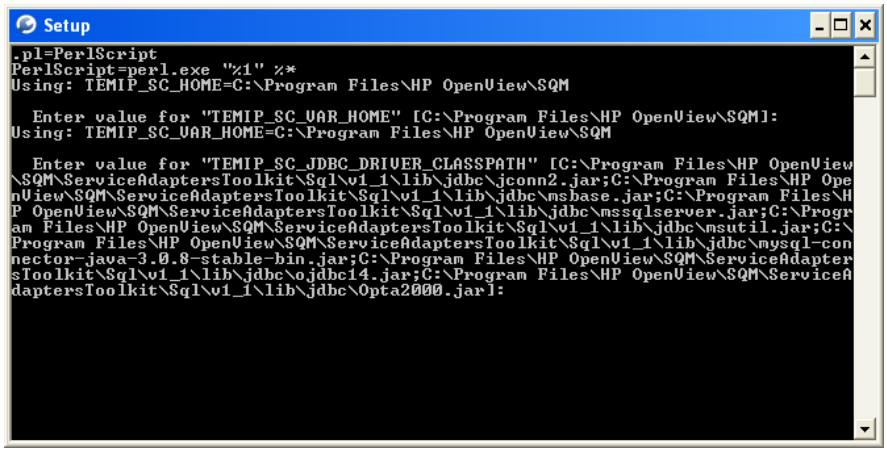
The given directory must exist and have the proper access rights (rw for the user).



The TEMIP_SC_JDBC_DRIVER_CLASSPATH environment variable which contains the list of JDBC driver paths.

If the user has copied all the drivers in the directory: **<SQM Installation directory>\ServiceAdaptersToolkit\Sql\v1_2\lib\jdbc,** the setup will prompt the list of discovered JDBC drivers, and, by simply pressing "**Return**" key, the default value is used. If no driver exists in this directory the user will have to enter each driver full path separated by ";" (see example below).

To provide new drivers (after a previous setup), it is necessary to run again the setup. The previous TEMIP_SC_JDBC_DRIVER_CLASSPATH value is proposed. The user must copy/paste the existing variable value and add new driver paths separated by ";".




**Note**

Each driver path must be separated by "**;**".

Example:

If all the drivers have been installed in **C:\Program Files\database**, the **TEMIP_SC_JDBC_DRIVER_CLASSPATH** will be set to:

C:\Program Files\database\jconn2.jar;C:\Program Files\database\msbase.jar;C:\Program Files\database\mssqlserver.jar;C:\Program Files\database\msutil.jar;C:\Program Files\database\ojdbc14.jar;C:\Program Files\database\Opta2000.jar

- No '"'character must be inserted in the JDBC Drivers CLASSPATH on Windows.

### On Unix

Connect as "**root**" user

Load the SQM environment variables (/var/opt/OV/SQM/slmv12/temip_sc_env.sh)

Perform the following commands:

```
# cd
$TEMIP SC HOME/ServiceAdaptersToolkit/Sql/v1 2/bin

# ./temip_sc_setup_sqltk.sh
```

The setup command will prompt the user to set:

The TEMIP_SC_VAR_HOME environment variable if it is not defined. The corresponding directory is used by the SQL SA Toolkit for logging and tracing.

**Note**

The given directory must exist and have the proper access rights (rw for the user).

The TEMIP_SC_JDBC_DRIVER_CLASSPATH environment variable which contains the list of JDBC driver paths.

If the user has copied all the drivers in the directory: **<SQM Installation directory>/ServiceAdaptersToolkit/Sql/v1_2/lib/jdbc,** the setup will prompt the list of discovered JDBC drivers, and, by simply pressing "**Return**" key, the default value is used.

If no driver exists in this directory the user will have to enter each driver full path separated by "**:**" (see example below).

Example:

If all the drivers have been installed in **/opt/database**, the **TEMIP_SC_JDBC_DRIVER_CLASSPATH** will be set to:

/opt/database/jconn2.jar:/opt/database/msbase.jar:/opt/database/mssqlserver.jar:/opt/database/msutil.jar:/opt/database/ojdbc14.jar:/opt/database/Opta2000.jar

**Note**

If additional drivers have to be supported, the SQL SA Toolkit setup needs to be performed again. A new setup will override the previous JDBC Driver CLASSPATH. If previous drivers still need to be supported, the user needs copy/paste the old CLASSPATH and add the new drivers path.

### 3.2.1 Configuring the SQL SA Toolkit license

The SQM utility to manage the licenses is ***temip_sc_license***.

To run the SQL SA Toolkit, it is necessary to configure the **SQM-SQL_SA_TOOLKIT** license.

At Kernel installation, a temporary **SQM-SQL_SA_TOOLKIT** license is installed. This temporary license allows a 90 days trial period and is activated on the first license check (when the application is started the first time).

Once Autopass is installed (in the same kit as the SQM Kernel) and SQL SA Toolkit has been configured, perform the following commands to check that the licenses have been correctly setup.

#### On Windows:

Open a command line and execute:

```
cd %TEMIP_SC_HOME%\bin
# To check that the license has been setup
temip_sc_license -check
```

#### On Unix:

Connect as "**root**" user

Set the TEMIP_SC_HOME environment variable, and check the license:

```
# export TEMIP SC HOME=<SQM installation directory>
cd %TEMIP_SC_HOME%\bin
temip_sc_license -check
```

When this temporary license key has expired, the SQL SA Toolkit cannot start anymore. A permanent license is then required. Please refer to the ***hp Openview SQM Administration Guide*** where is explained how to retrieve SQM licenses and import them into Autopass (with ***temip_sc_license*** utility).

## 3.3 Starting the SQL SA Toolkit

To start the SQL SA Toolkit, the following commands have to be executed.

### On Windows

- `Select menu: All Programs -> HP OpenView -> SQM -> Sql SA Toolkit v1_2 -> Launch`

### On Unix

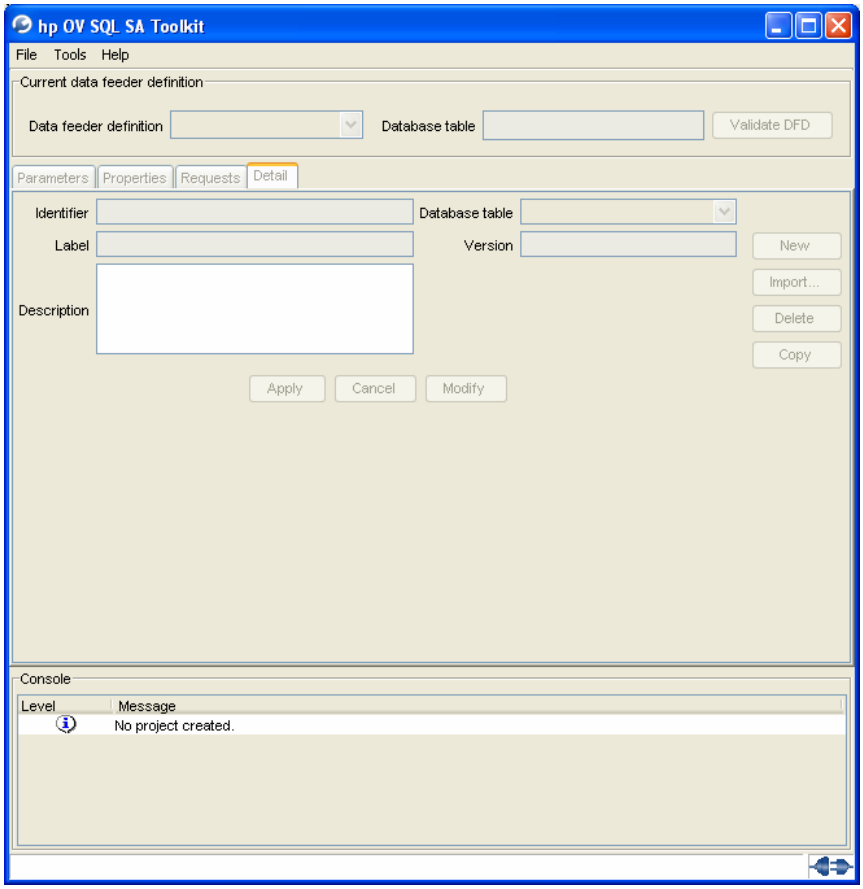
Connect as “sqmadm” user

Load the SQM environment variables
(/var/opt/OV/SQM/slmv12/temip_sc_env.sh)

Perform the following commands:

```
# export DISPLAY=<display>
# cd $TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin
# ./temip_sc_start_sqltk.sh
```

The SQL SA Toolkit main window appears and the customization steps can start.

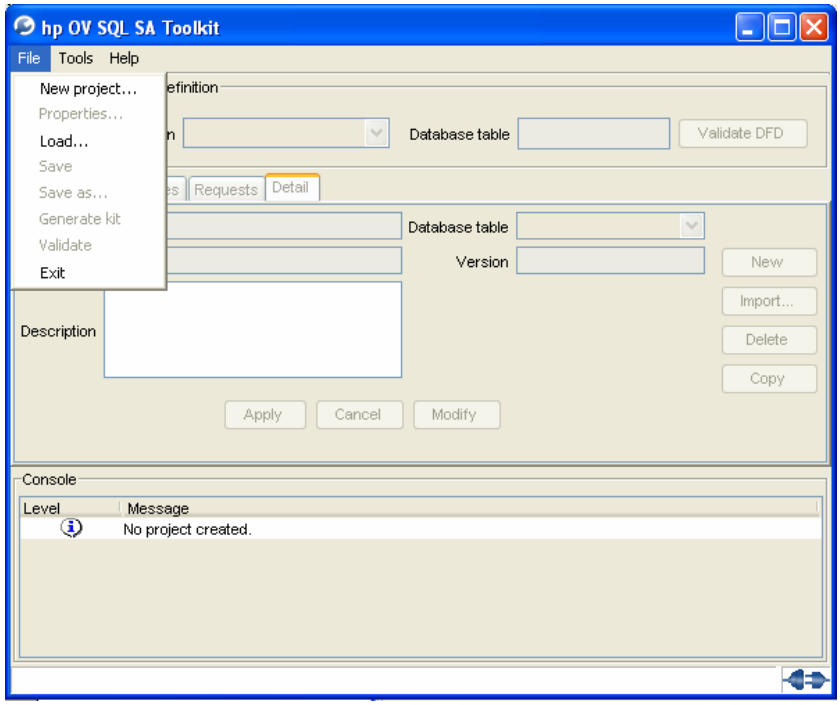


## 3.4 Creating a customization project

The definition of a customization project is the first mandatory step when using the SQL SA Toolkit.

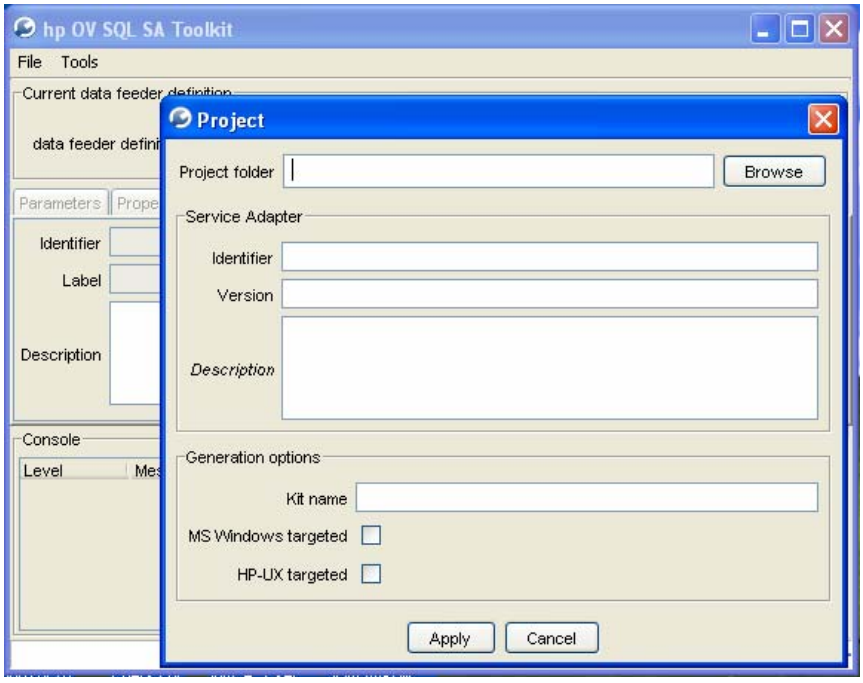
By selecting the **File** menu from the main window, the following project submenus appear:

- **New Project…** (to enter a new project characteristics).

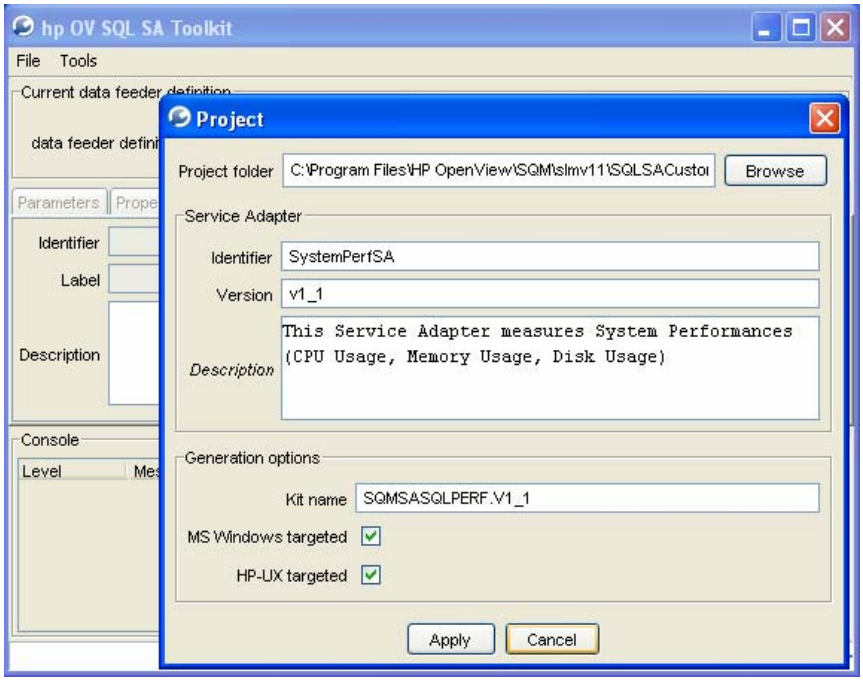



- **Load…** (to load an existing project workspace).

Choosing the option **New Project…** will open the Project characteristic window below to collect the project inputs:

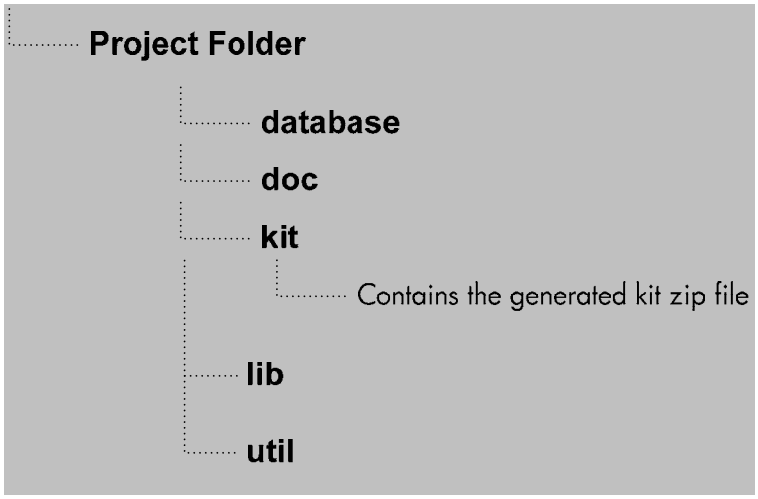



In the **Project Window**, the user will have to provide:




- *Project* Folder: Folder where the project source files will be placed, and the kit will be generated.
- Service Adapter
- *Identifier*: Name of the SQL Service Adapter customization (SA Name)
- *Version*: Version of the SQL Service Adapter customization. The user may want to define incremental versions of the same Service Adapter with different mappings. The version syntax is: *v<major version>_<minor version>*.
- Ex: v1_0.
- *Description*: brief description of the SQL Service Adapter customization
- Generation options:
- *Kit name*: Name of the zip file that will be generated by the customization. The name is defined by the user, and it is recommended that the name contains the Service Adapter Name and version (to avoid any confusion). Avoid using space characters in the kit name.
- *Targeted platform*: Platform on which the SQL Service Adapter customization will run. If both platforms are selected, in this case, the generated kit will be installable on MS Windows and HP-Unix.

Once the customization project is created, the following project directory layout is initialized by the SQL SA Toolkit in the project directory mentioned by the user.




To modify the Project Folder location, the user may select the menu:

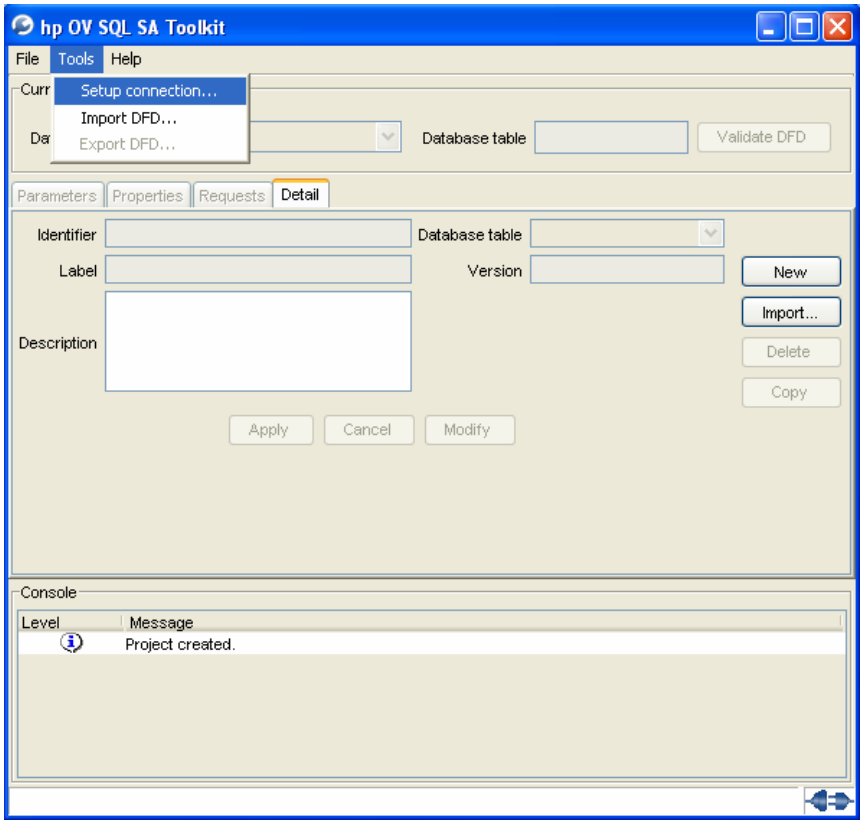
```
File -> Properties
```

This action will open the project definition window and provide access to the project characteristics.

## 3.5 Creating a connection

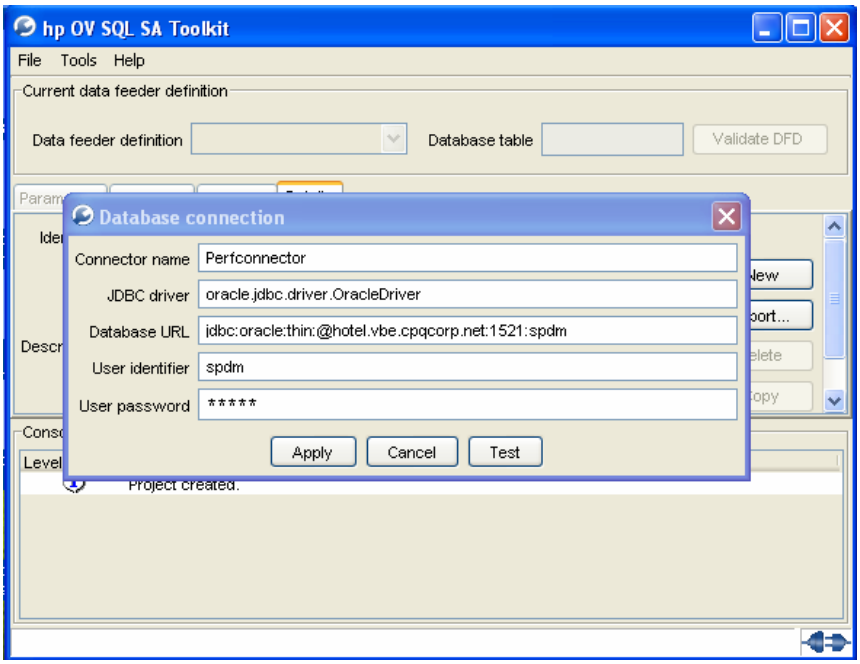
The database connection parameters can be defined by selecting the menu:

```
Tools -> Setup Connection
```



- The connection window appears.

The user will enter the following connection characteristics:




- *Connector name*: the name that will be used by the SQL Service Adapter to store the connection parameters
- *JDBC driver*: should be the driver that has been installed to connect to the database
- *Database URL*: A JDBC URL provides a way of identifying a data source so that the appropriate driver will recognize it and establish a connection with it.

The standard syntax for JDBC URLs is shown here. It has three parts, which are separated by colons.

jdbc:<subprotocol>:<subname>

The three parts of a JDBC URL are broken down as follows:

The protocol in a JDBC URL is always jdbc

<subprotocol>-the name of the driver or the name of a database connectivity mechanism, which may be supported by one or more drivers

<subname>-a way to identify the data source. In our example, it is the system name and port number.

- *User identifier*: login name to access the database
- *User password*: associated login password.
- You need to make sure, that the chosen connection user has enough rights to view the database tables.

Here are examples of connection characteristics adapted to the previously recommended JDBC drivers:

```
Oracle 9i (8i):
 DriverName:      oracle.jdbc.driver.OracleDriver
 Url:       jdbc:oracle:thin:@host.domaint:port:mydb
Sybase:
DriverName:       com.sybase.jdbc2.jdbc.SybDriver
Url:  jdbc:sybase:Tds:host.domain:port/mydb
 Microsoft SQL Server:
 DriverName:      com.microsoft.jdbc.sqlserver.SQLServerDriver
Url:jdbc:microsoft:sqlserver://host.domain:port;DatabaseName=mydb
Please refer to Appendix B for SQL database configuration
requirements.
```

Using the **Test** button, the SQL Service Adapter Toolkit will try to connect to the database. The result of the test will be displayed in the console. If any error returned, please check your connection parameters.

By Using **Apply** button, the SQL SA Toolkit will try to retrieve SQL table information from the database.

When the connection is successful, the connected mode is identified by the icon  at the right bottom of the window.

If the database is not accessible, the connector information will be saved, but the next Data Feeder modeling phase will have to be done without the table/view columns information (the table/view column will be entered manually by the user and is not prompted automatically). This is called *offline* mode.

## 3.6 Modeling/Mapping a DFD

The Data Feeder modeling and mapping phase starts with the definition of the Data Feeder characteristics: Name, Identifier, Version and Description.

These Data Feeder definition characteristics can be retrieved:

- By loading an existing DFD definition that has been generated from Service Designer user interface
- By prompting the user to enter the DFD characteristics.

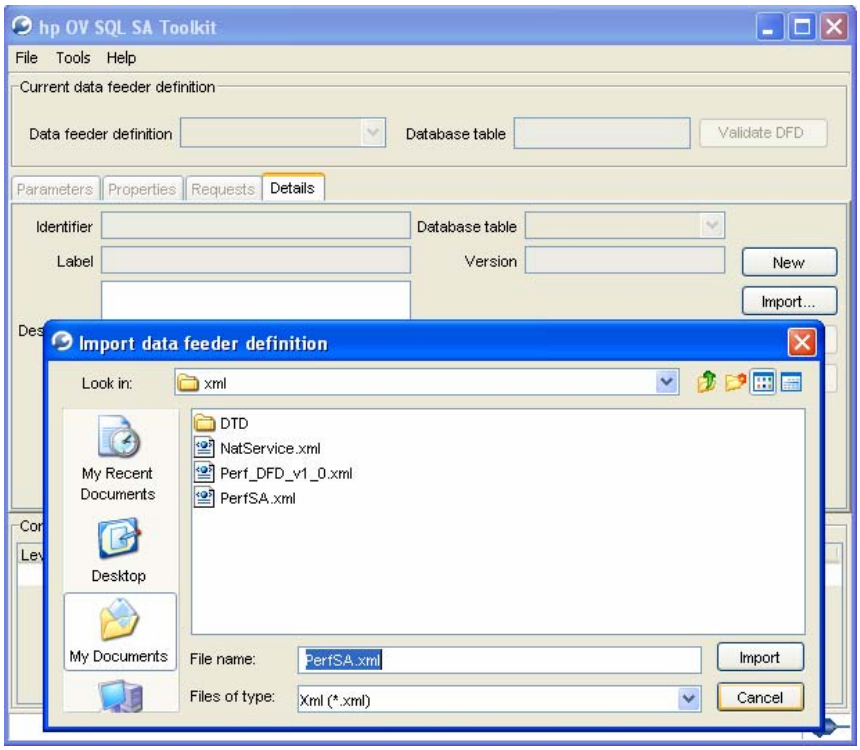
Both ways are presented within the following chapters.

### 3.6.1 Importing a DFD from a Service Designer XML file

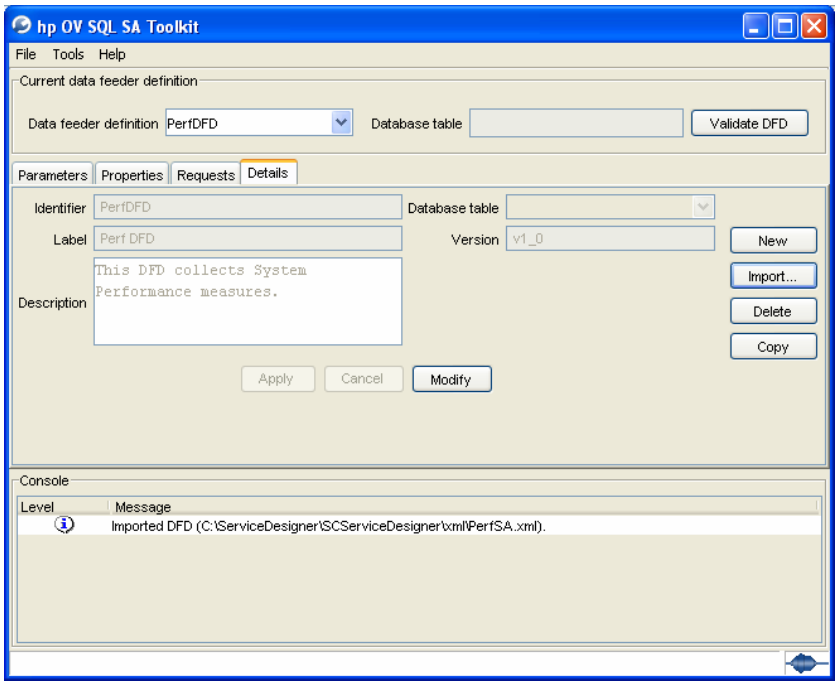
If the user has previously designed a Service Model and associated data feeders using the Service Designer, the toolkit is able to import the generated data feeder XML definition.

The following steps need to be performed:

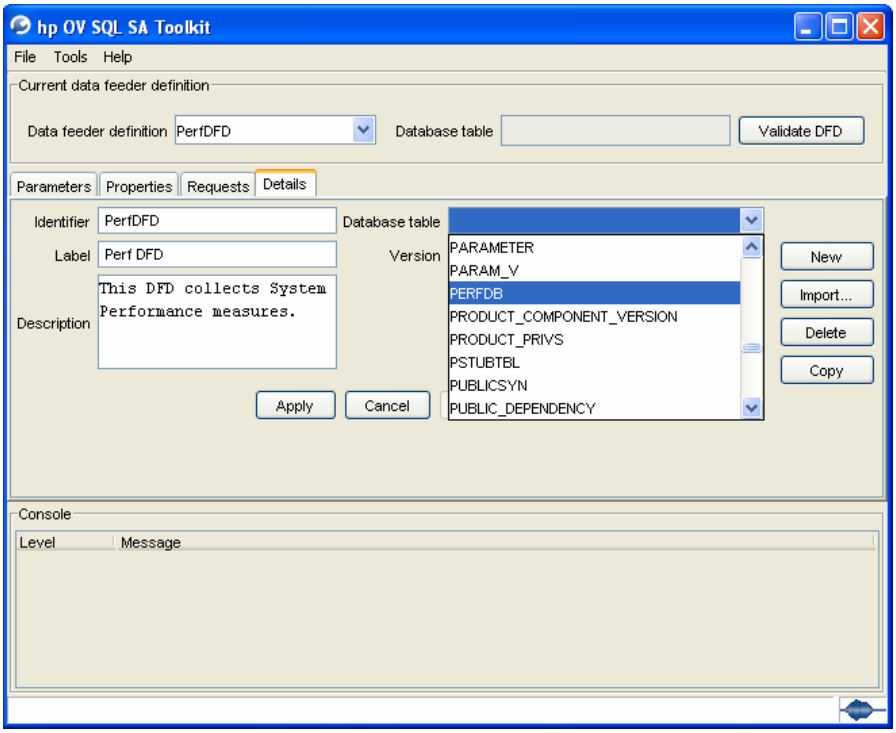
From the **Details** tab, select the **Import…** button. A file browser is opened to locate the DFD XML definition file.




Select **Import** button to load the file.



1. Click on **Modify** button to edit the DFD definition and select **Database table** name (or view) which will be used for the mapping (the list of available table or views is only available if the connection to the database was successful, otherwise, the user will enter the table name manually)




Once the Data Feeder definition file is loaded, the parameter mapping can start (refer to the **Mapping the DFD parameters** section).

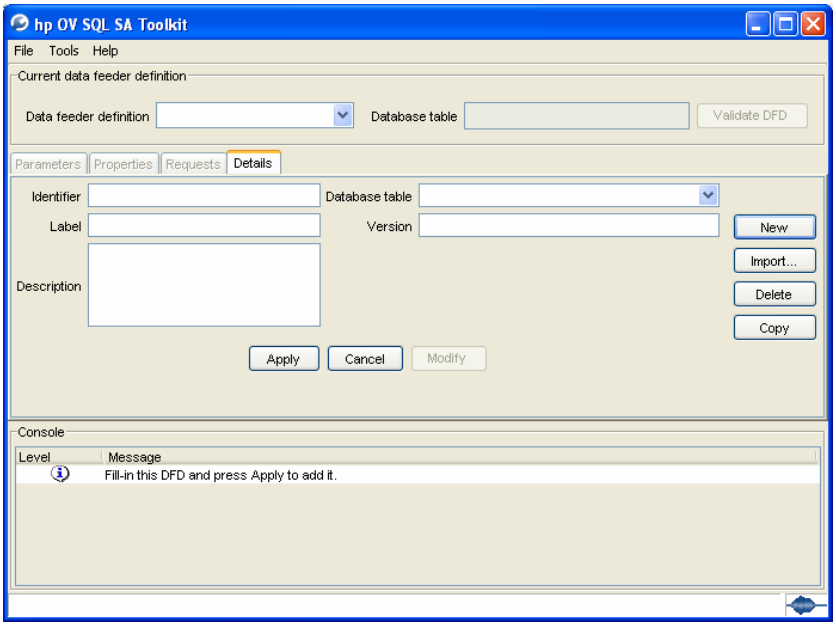
**Important Note**

- If the user has defined a specific MRP Naming Schema containing properties and fixed strings (in SQM Service Designer), the *import DFD XML* definition will remove from the MRP Naming Schema the fixed strings, and the generated DFD will not contain these strings anymore (only the properties that are part of the MRP will be restituted in the DFD definition).
- The SQL SA Toolkit does not offer the possibility to define SQM Enumeration parameter data type from the User Interface. It is recommended to define the SQM Enumeration parameters from the Service Designer and use the Import feature to load the Enumerate definition into the SQL SA Toolkit.
- An **Export** feature is also available from the Toolkit User Interface (`Tools -> Export DFD`), it the user needs to export a DFD to work in Service Designer, and re-import it after.

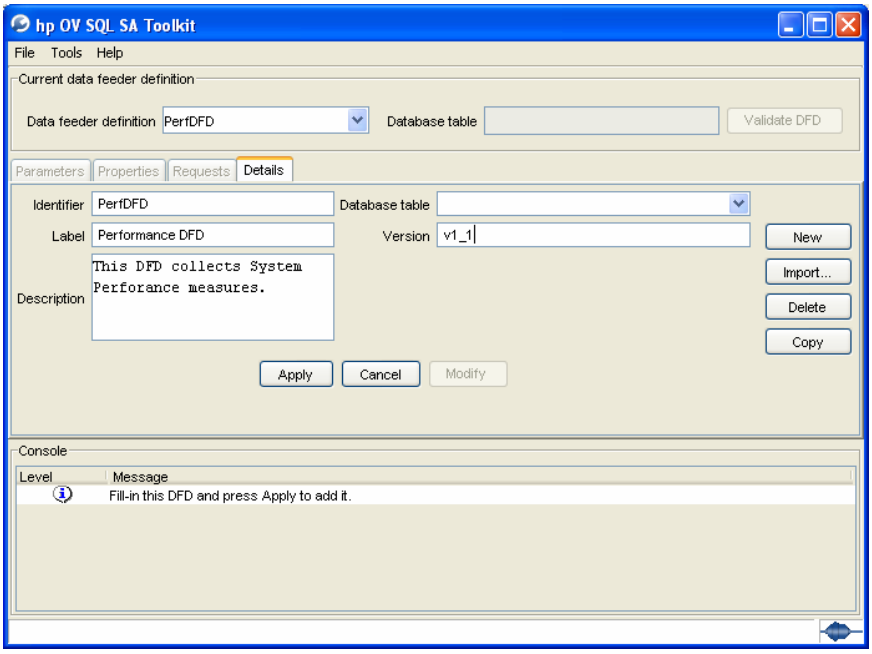
## 3.6.2 Defining a Data Feeder using the Toolkit

If the user does not have an existing data feeder XML definition file, he has to manually enter all the DFD characteristics using the following steps:

1. Select the **Details** tab in the main window.




2. Click **New** then fill the **Details** panel to provide the DFD characteristics:

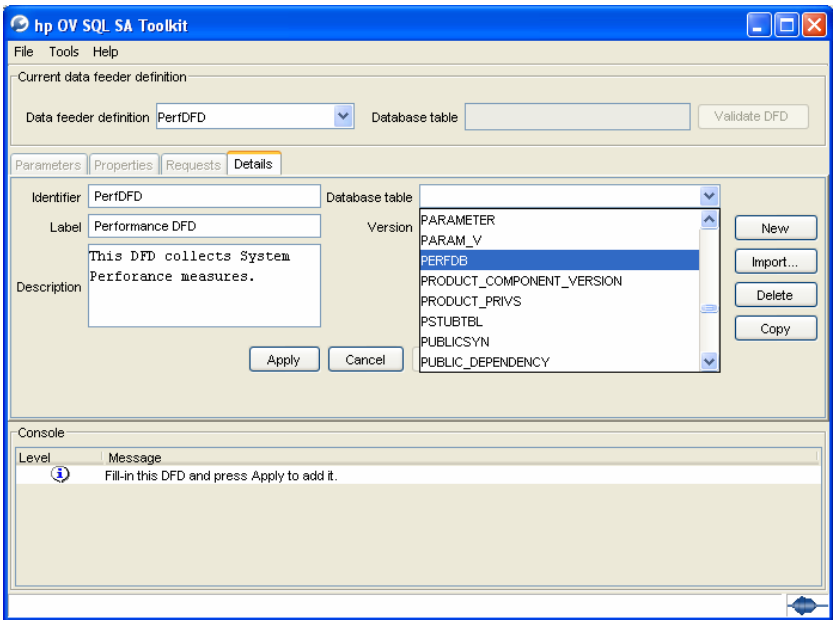



The following DFD characteristics have to be entered:

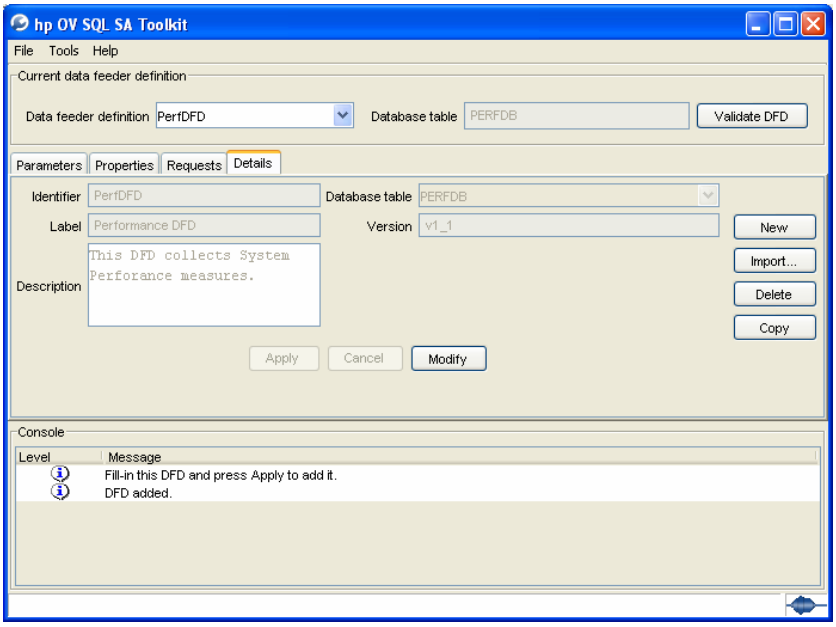
- *Identifier*: The DFD identifier which cannot exceed 16 characters.
- *Label*: display name of the DFD.
- *Description*: a brief description of the DFD.
- *Database table*: SQL table or view associated to the DFD. This table or view will be used to collect the DFD parameter values.

- Version: DFD version. The version syntax is: v<major version>_<minor version>.Ex: v1_0.

3. If the connection to the database has been successful, the list of tables will be presented, otherwise, the user will enter the table name manually.



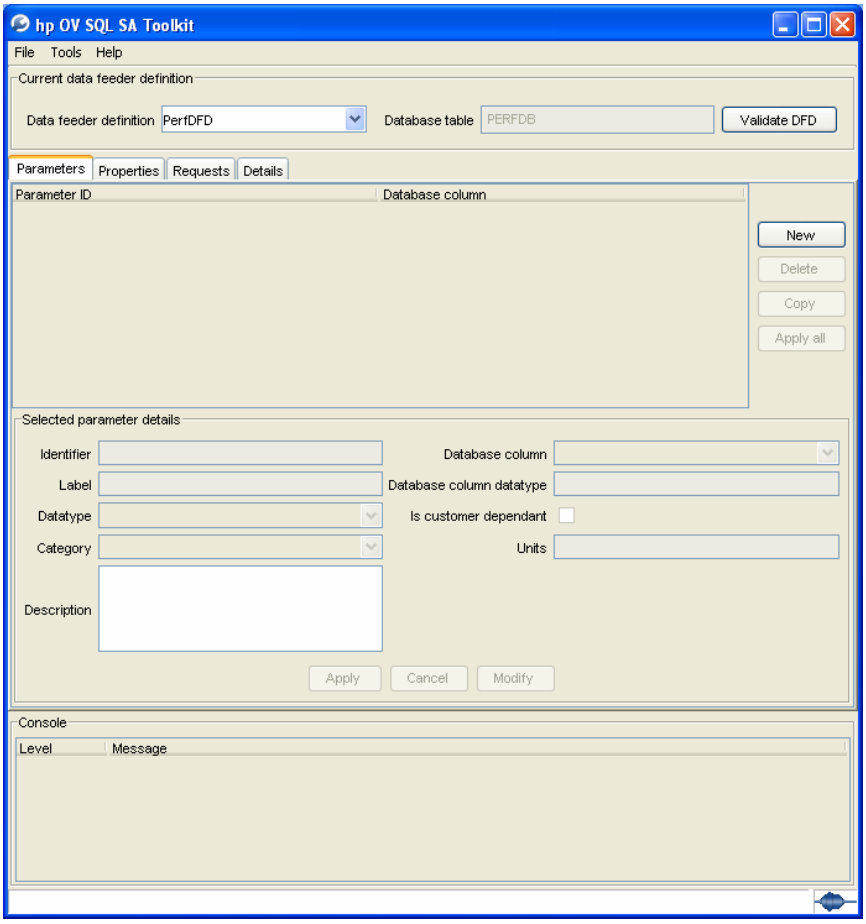

4. Click on **Apply** button to accept the DFD details. Then the other Toolkit tabs will be accessible for DFD mapping.




## 3.6.3 Mapping the DFD parameters

The parameter mapping phase will allow the association of a Table/View column with a DFD parameter.

By clicking on the **Parameters** tab in the toolkit main window, the parameter mapping window will appear.




The following possibilities are available at the Data Feeder parameter mapping phase:

- **New**

By selecting the **New** button, a new parameter entry will be created in the table and the Parameter Detail window becomes editable.

- **Copy**

Selecting an existing parameter in the table and clicking on the **Copy** button will create the same parameter (with its associated mapping) in the table. The parameter copy has to be edited and modified.

- **Delete**

Selecting a parameter in the table and clicking on the **Delete** button will remove the parameter from the Data Feeder definition (along with the associated mapping).

- **Apply all**

The "**Apply all"** button, will apply all modifications performed on the parameters listed in the table. Parameters, for which modifications have not been applied yet, are presented in grey color.

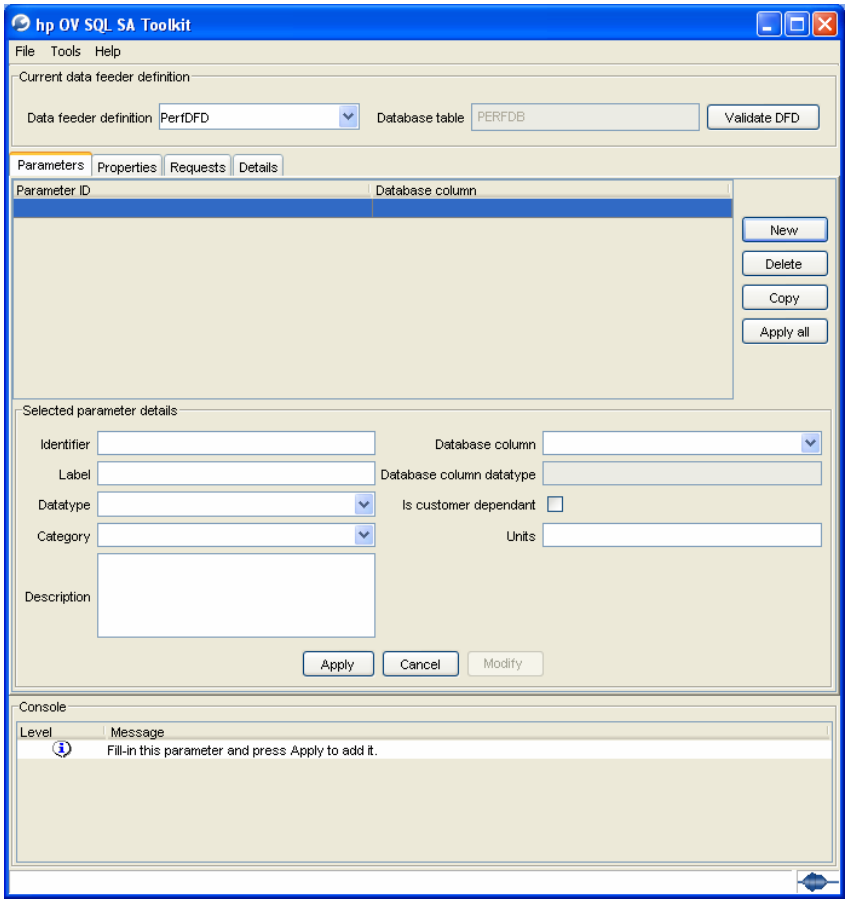
For each parameter defined, once selected, the user has access to the mapping information in the **Selected parameter details** panel (lower part of the window) by clicking on the "**Modify**" button. The user can modify the mapping

information (change the database column associated to the Data Feeder parameter and modify the parameter data type, category, units, description or identifier). Once the modification has been performed, click on **Apply** button to confirm the parameter mapping modification.

If the connection to the database is available (this is determined at the connection definition phase), the user interface will prompt the existing table/view columns in the database.

Here are the steps to create, edit and modify a parameter mapping:

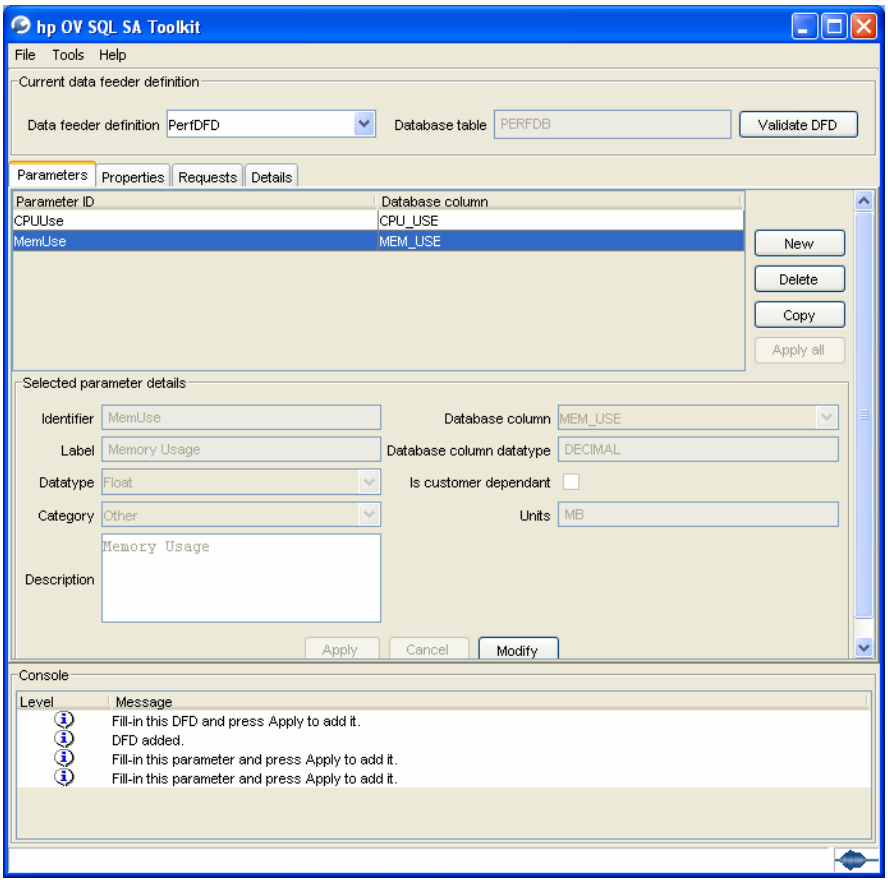
1. Click on **New** button to create and edit a parameter.




2. Fill in the ***Selected parameter details*** pane with:

- *Identifier*: DFD parameter identifier (limited to 16 characters).
- *Label:* DFD parameter label.
- *Datatype:* parameter data type which can be one of the SQM available data types (String, Int, Float, AbsTime, RelativeTime).
- *Category:* parameter category which can be one of the SQM available categories (Rate, Percent, Counter, Gauge, Other).
- *Description:* brief description of the parameter.
- *Database column*: the SQL database column from which the parameter values will be collected. This column name can be chosen among available table columns if the connection to the database could be done, otherwise the user will manually enter the table column name.

- *Database column datatype*: SQL column data type (retrieved from the SQL database).
- *Is customer dependant*: indicates if the parameter is customer (subscriber) dependant or not. If the parameter is customer dependant, at the collection request definition, the Customer or Subscriber information will need to be provided (refer to section **Defining the collection request**)
- *Units*: parameter unit. The unit is used at the SQM User interface (SLA Monitoring) to display the parameter value information.




3. Click **Apply** to take into account the modifications.

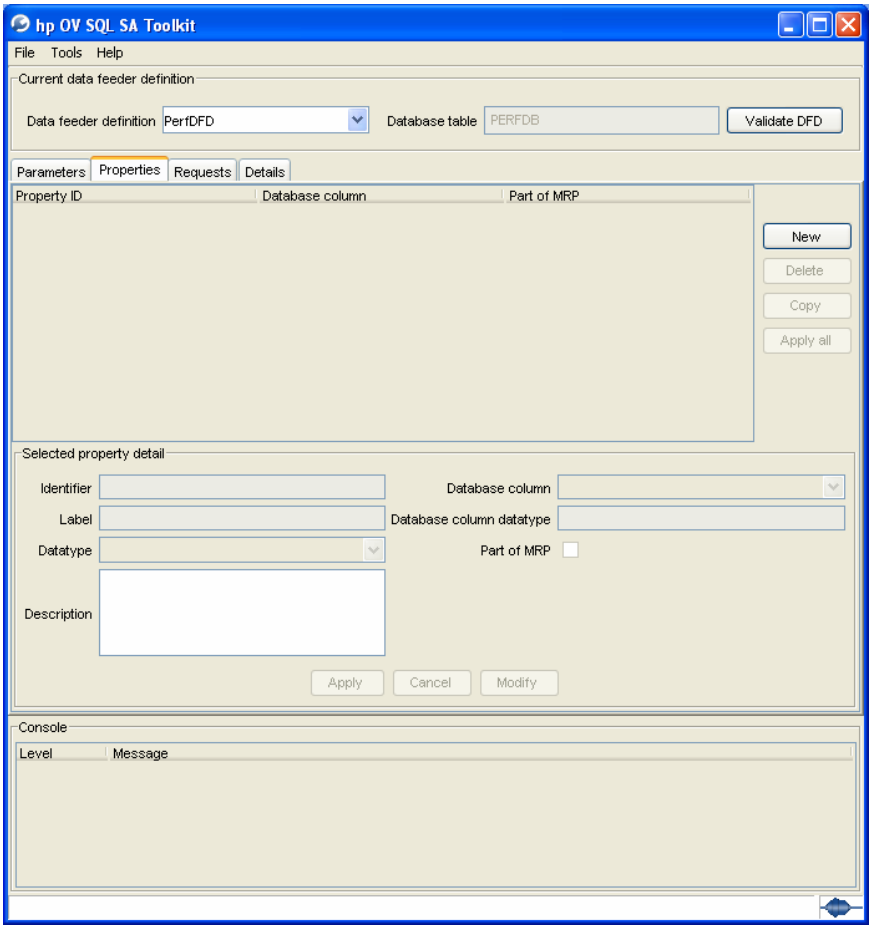
The same processing needs to be done for each parameter that is to be mapped to a table column.

## 3.6.4 Mapping the DFD properties

The property mapping phase will allow the association of a Table/View column to a DFD property.

By clicking on the **Properties** tab in the toolkit main window, the property mapping window will appear.

The following possibilities are available at the Data Feeder property mapping phase:




- **New**

By selecting the **New** button, a new property entry will be created in the table.

- **Copy**

Selecting an existing property in the table and clicking on the **Copy** button will create the same property (along with its associated mapping) in the table. The property copy must be edited and modified.

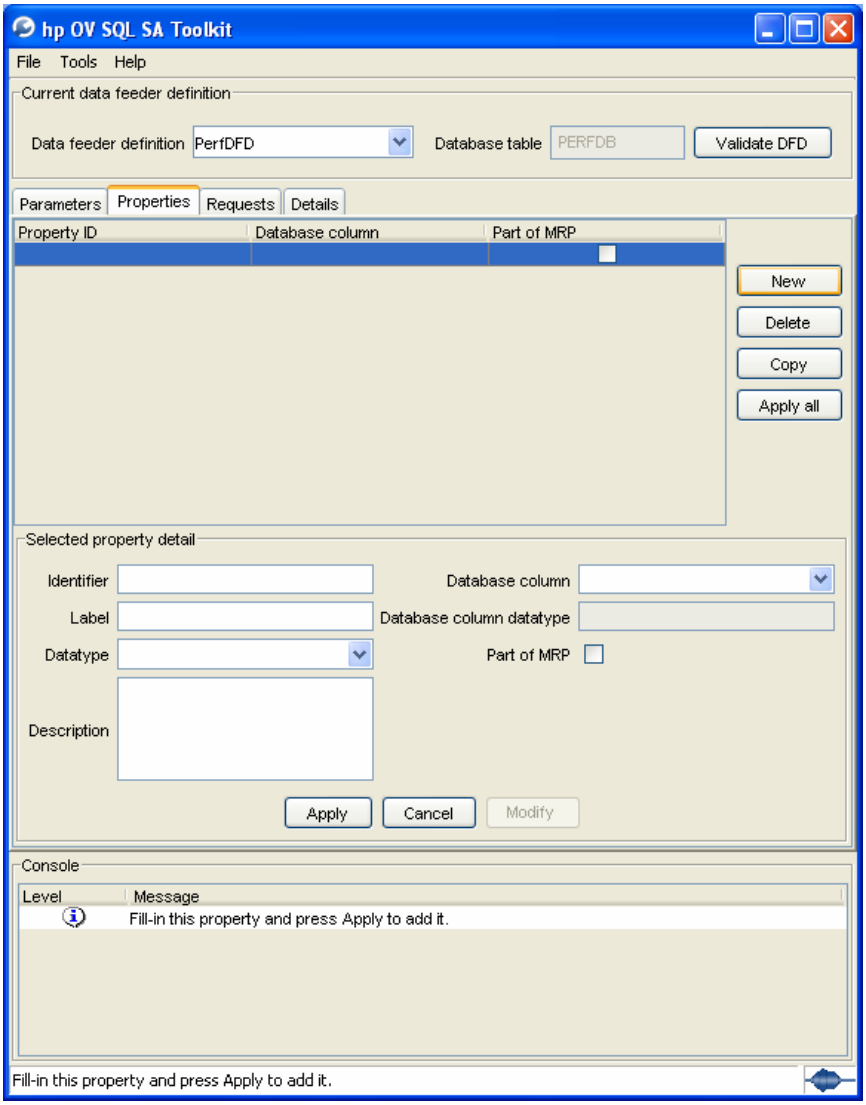
- **Delete**

Selecting a property in the table and clicking on the **Delete** button will remove the property from the Data Feeder definition (along with the associated mapping).

- **Apply all**

The "**Apply All"** button, will apply all modifications performed on the properties listed in the table. Properties, for which modifications have not been applied, are presented in grey color.

For each property defined, once selected, the user has access to the mapping information in the **Selected property details** pane (lower part of the window) by clicking on the "**Modify**" button. The user can modify the mapping information (change the database column associated to the Data Feeder property and modify the property data type, description or identifier, and decide whether this property is part of the DFD MRP or not).




**Note**

The user has to insert the properties according to their order in the MRP Naming Schema.

If the connection to the database is available (this is determined at the connection definition phase), the user interface will prompt the existing table/view columns.

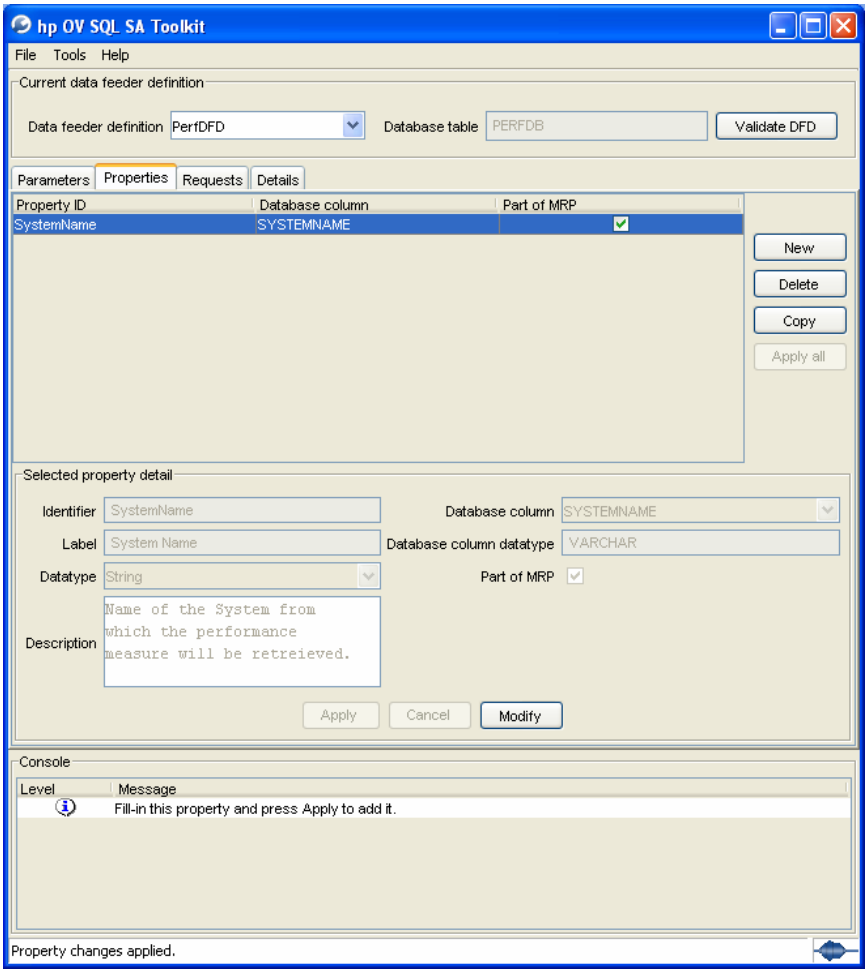
Here are the steps to create, edit and modify a property mapping:

1. Click on **New** button to create and edit a property.

2. Fill in the **Selected property details** pane with:

- *Identifier*: DFD property identifier (limited to 16 characters).

- *Label*: DFD property label.
- *Datatype:* property data type which can be one of the SQM available data




  types (String, Int, Float, AbsTime, RelativeTime).
- *Category:* parameter category which can e one of the SQM available categories (Rate, Percent, Counter, Gauge, Other).
- *Description:* brief description of the parameter.
- *Database column*: the SQL database column from which the property value will be retrieved. This column name can be chosen among available table columns if the connection to the database could be done, otherwise the user will manually enter the table column name.
- *Database column datatype*: SQL column data type (retrieved from the SQL database)
- *Part of MRP*: indicates if the property is part of the MRP or not.

3. Click **Apply** to take the modifications into account.

The same processing has to be done for each property that is to be mapped to a table column.

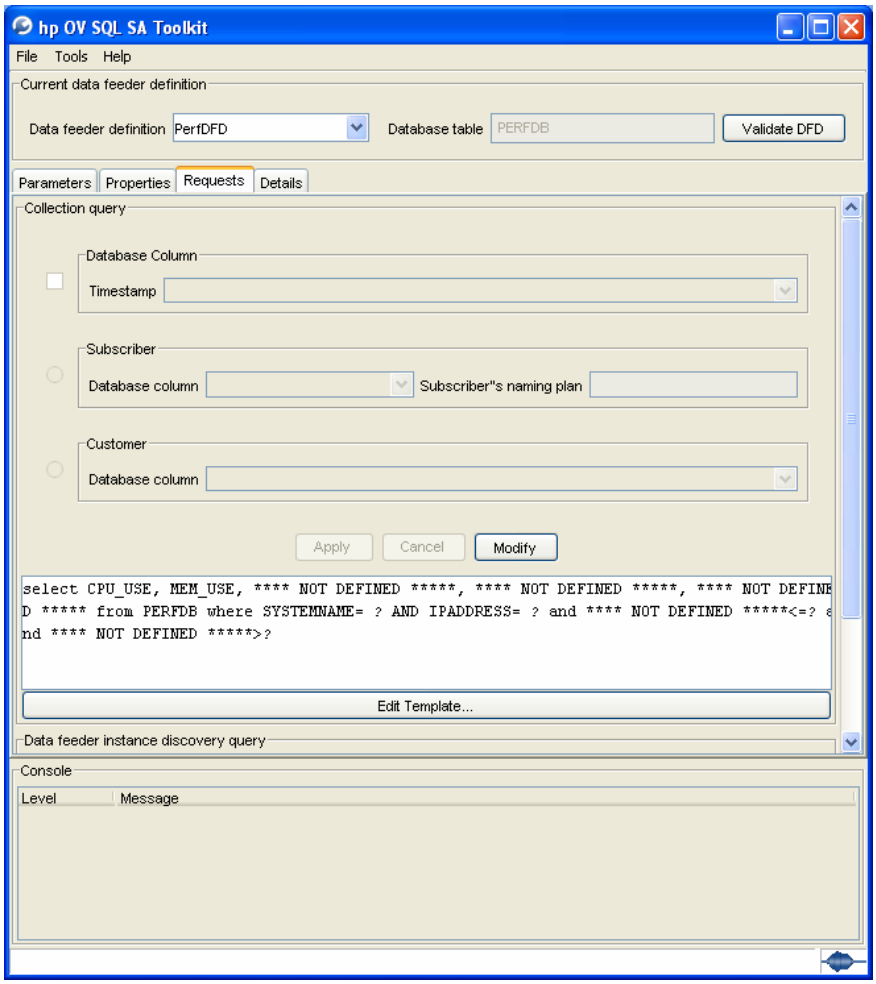
At the end of this phase, the user may choose to check if the mapping is correct, using the **Validate DFD** button. This function checks that all parameters are correctly mapped to a column name, and if all mandatory parameter fields are presents for future load onto the SQM repository.

This **Validate DFD** will not validate the mapping against the database (it will not validate that a specified column in the mapping definition exists in the table/view).

## 3.7 Defining the collection request

At this step, the user defines the SQL request that will be performed to the database for collecting the Data Feeder parameter values (collection request).

By selecting the **Requests** tab in the main toolkit window, the request configuration window appears.

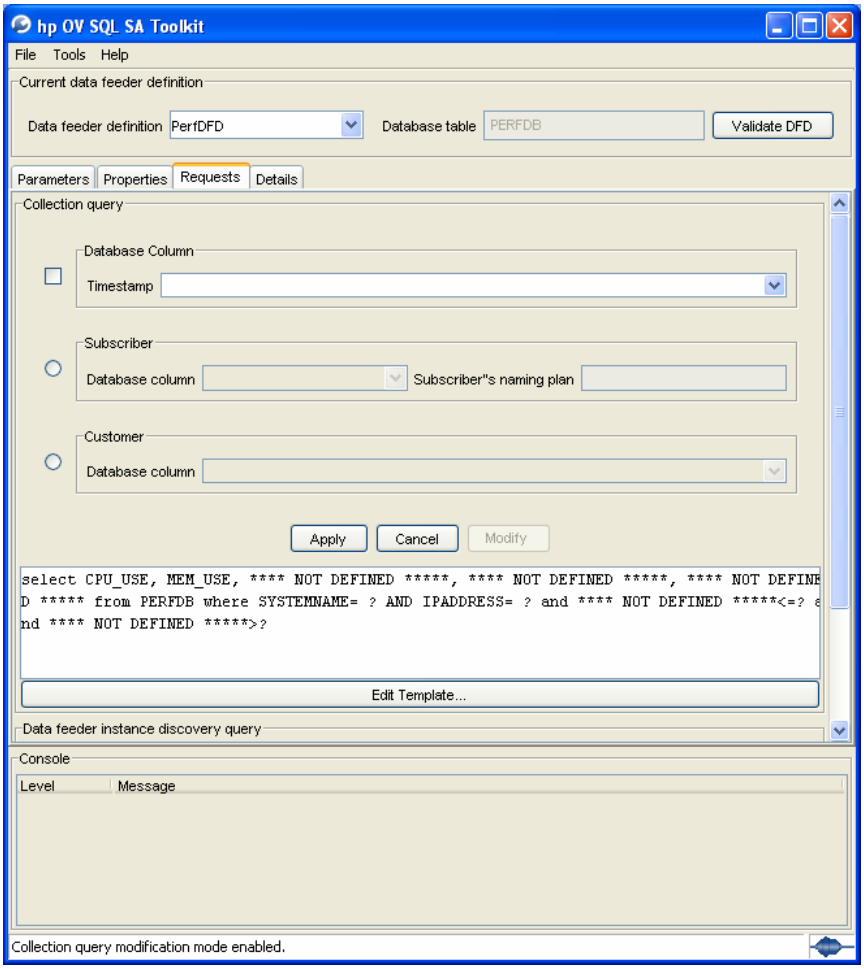



Before customizing the collection request, the user needs to identify the timestamp column (if it exists), and the Subscriber or Customer column if the DFD parameters are customer dependant.

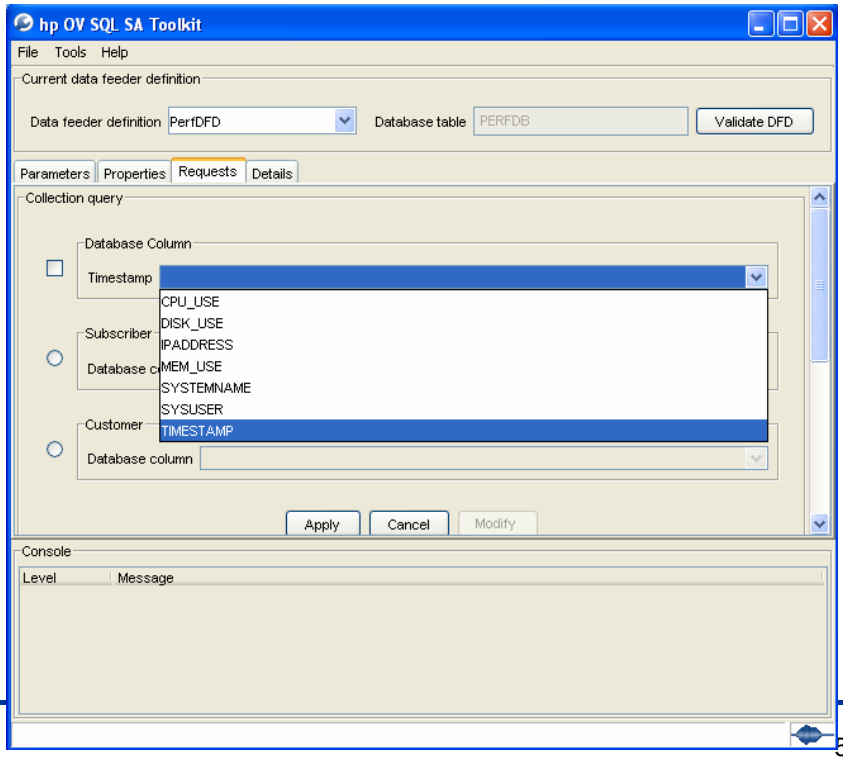
### 3.7.1 Mapping the timestamp column

If the SQL database associated to the DFD contains a timestamp column, it is necessary to provide the mapping information to this column.

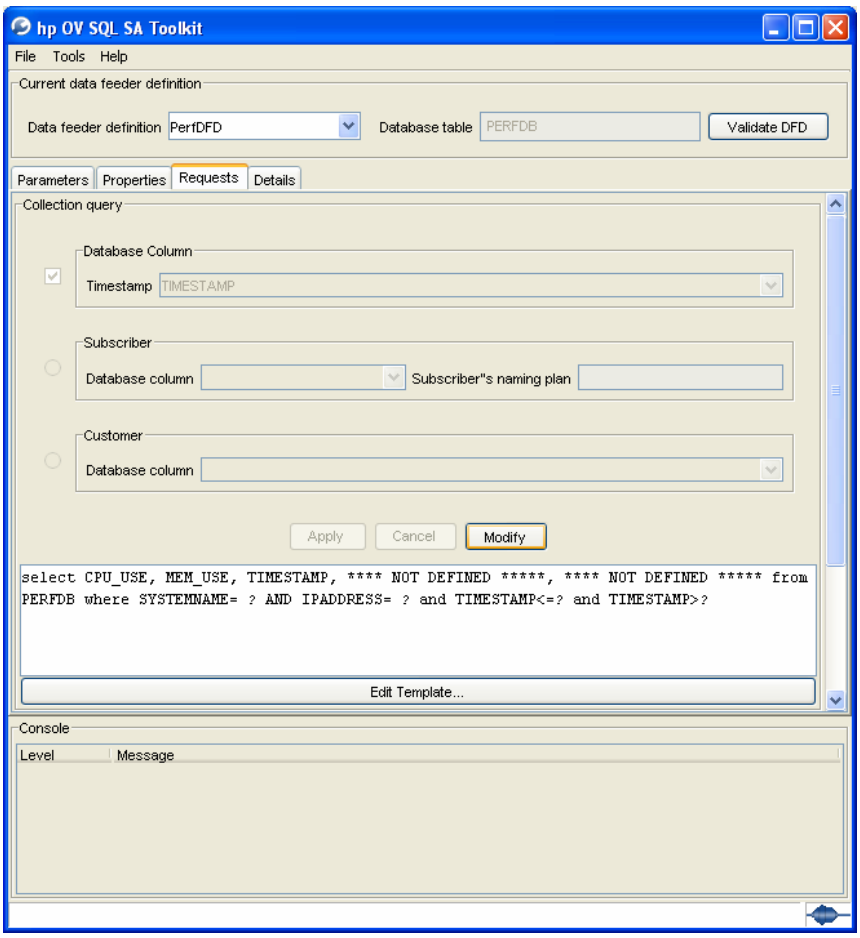
1. To do so, select the **Requests** tab, and, in the **Collection query** part, click on **Modify** to edit the Timestamp mapping.




2. Select the name of the database column which contains the timestamp information. If the Toolkit is not connected to the database, enter manually the column name.



3. The Timestamp check box is automatically selected.




4. Click **Apply** button to accept changes. The collection query is updated.

**Important Note**

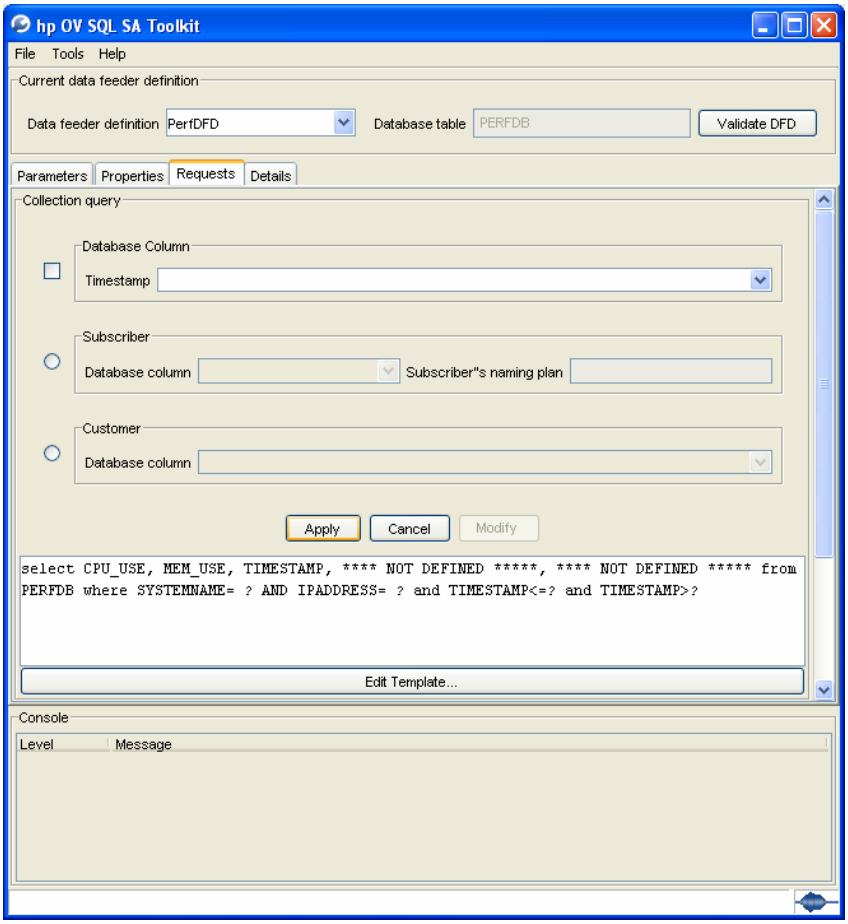
If **no timestamp column is available** in the database, the previous actions will not be performed. It is important to edit the default collection query and remove any reference to the **[TIMESTAMP]** keyword (in the **SELECT** and **WHERE** clauses of the collection request. The user will have to find the appropriate **WHERE** clause that will return the expected parameter values at the SQL SA polling period.

## 3.7.2 Mapping customer or subscriber

As previously explained in section **Definitions,** if the DFD parameters are customer dependant, it is necessary to associate to the collected values, the customer or subscriber information. Such information has to exist in the database as a table column. This section explains how to map this information.

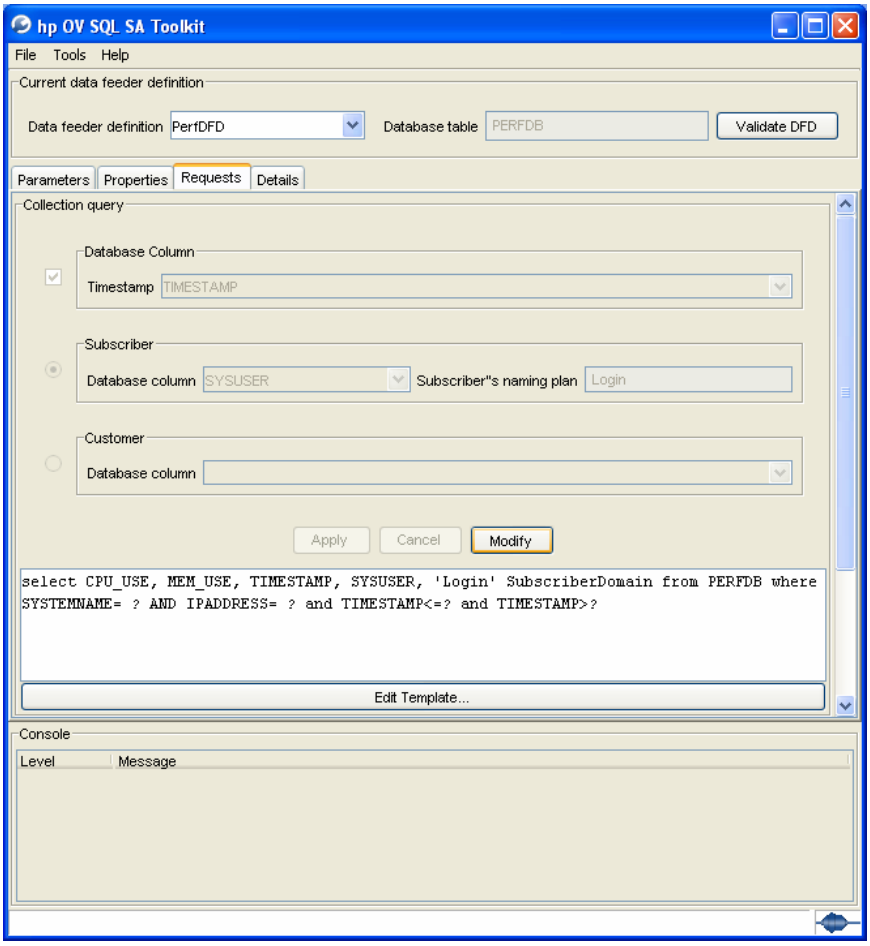
### 3.7.2.1 Mapping a subscriber

1. Select the **Requests** tab, and in the **Collection query** part, click on **Modify** to edit the Subscriber or Customer mappings.



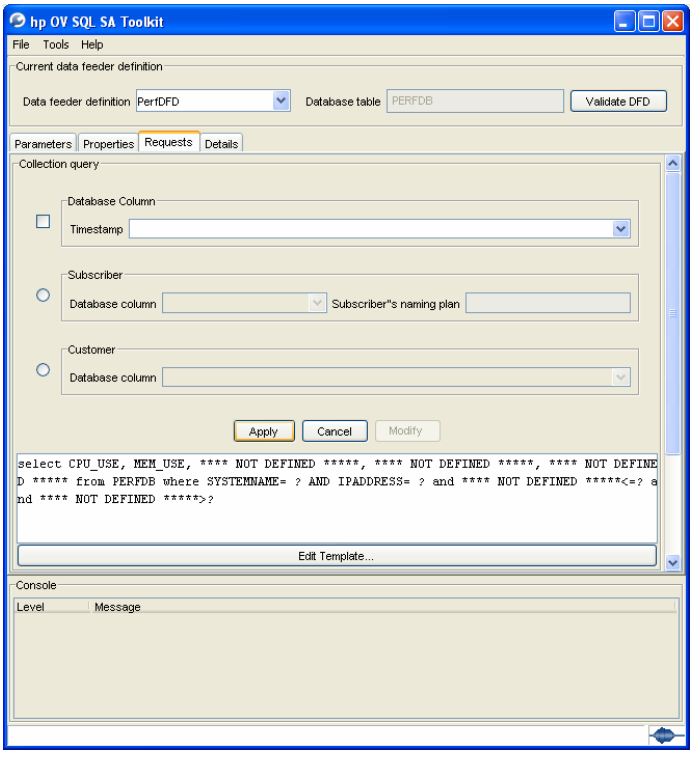

2. If the database contains a subscriber ID column, select in the Subscriber pane, the subscriber table column. Then provide the naming domain (*Subscriber's naming plan*) that will permit to associate the subscriber ID to a customer ID. The *Subscriber's naming plan* is a fixed string. The Customer, Subscriber, Naming Plan definitions have been presented in the chapter **Definitions**. Please refer to this chapter and to the ***SQM Information Modeling Reference Guide*** for details about SQM customers and subscribers information.

3. Then click **Apply** to accept changes.

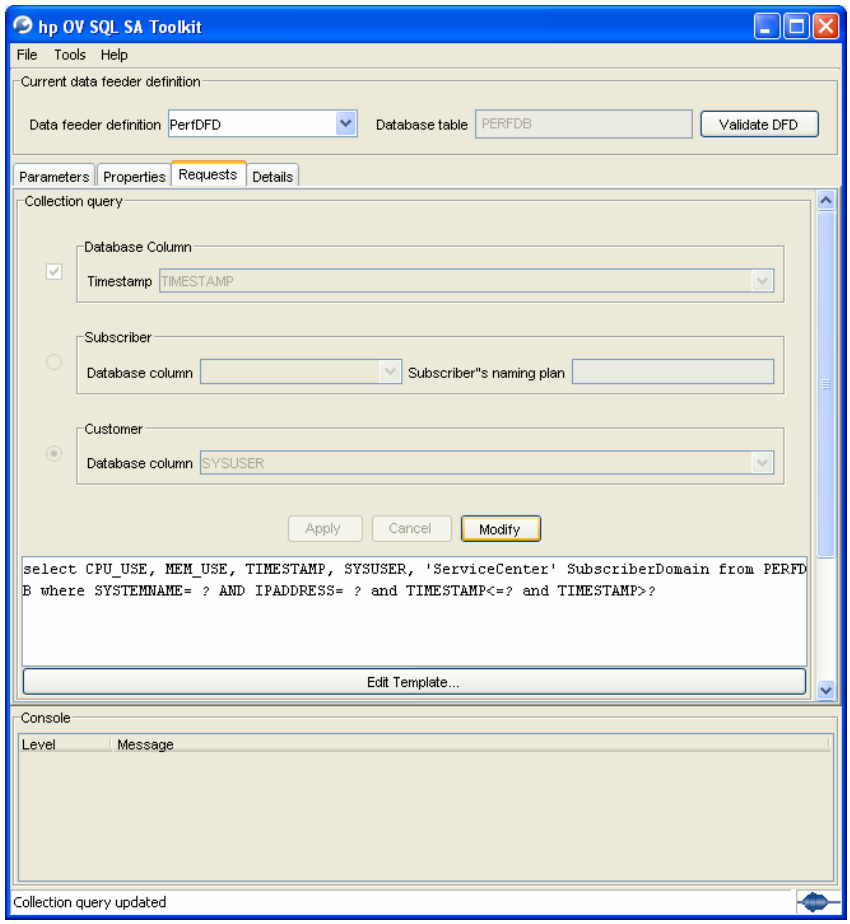



### 3.7.2.2 Mapping a customer

1. Select the **Requests** tab, and in the **Collection query** part, click on **Modify** to edit the Customer mapping.



2. If the database contains a customer ID column, select in the Customer pane, the customer table column.




3. Then click **Apply** button to accept changes.

## 3.7.3 Modifying the collection query

A standard SQL collection request is proposed to the user. This request contains some predefined keywords that are processed by the SQL Service Adapter.

The default request is as follows:

```
SELECT [PARAMETERS], [TIMESTAMP],[SUBSCRIBER],
[SUBSCRIBER_DOMAIN] FROM [TABLES] WHERE [MRP_PROPERTIES] AND
[TIMESTAMP]<=[CURRENT_GMT_TIME] AND
[TIMESTAMP]>=[LAST_TIMESTAMP]
```

This default request is valid if the database table or view associated to a DFD, has the standard columns presented in chapter **Definitions** (with Timestamp, Parameters, Customer or Subscriber, MRP property columns).

This request collects all DFD parameter raw data for specific MRPs, whose timestamp is between the current GMT time and the last collected timestamp value for the Data Feeder Instance.

**Important Note**

The default collection request has to be modified when the TIMESTAMP, the CUSTOMER or SUBSCRIBER columns are not available.

If there is no timestamp column, remove all references to the **[TIMESTAMP]** keyword, and provide a WHERE clause that returns the last inserted parameter values for a specific DFI.

If no customer or subscriber information is provided in the database, remove the keywords **[SUBSCRIBER]** and **[SUBSCRIBER_DOMAIN]** from the collection query, in the **SELECT** clause.

The following keywords can be used at the edition of the collection query.

For **SELECT** clause:

The following keywords are used in the select clause of the collection request template:

- **[PARAMETERS]**: will be replaced by the list of parameters defined in the DFD.
- **[TIMESTAMP]**: will be replaced by timestamp column name
- **[SUBSCRIBER]**: will be replaced by subscriber column name
- **[SUBSCRIBER_DOMAIN]** : will be replaced by subscriber domain column name

For **FROM** clause:

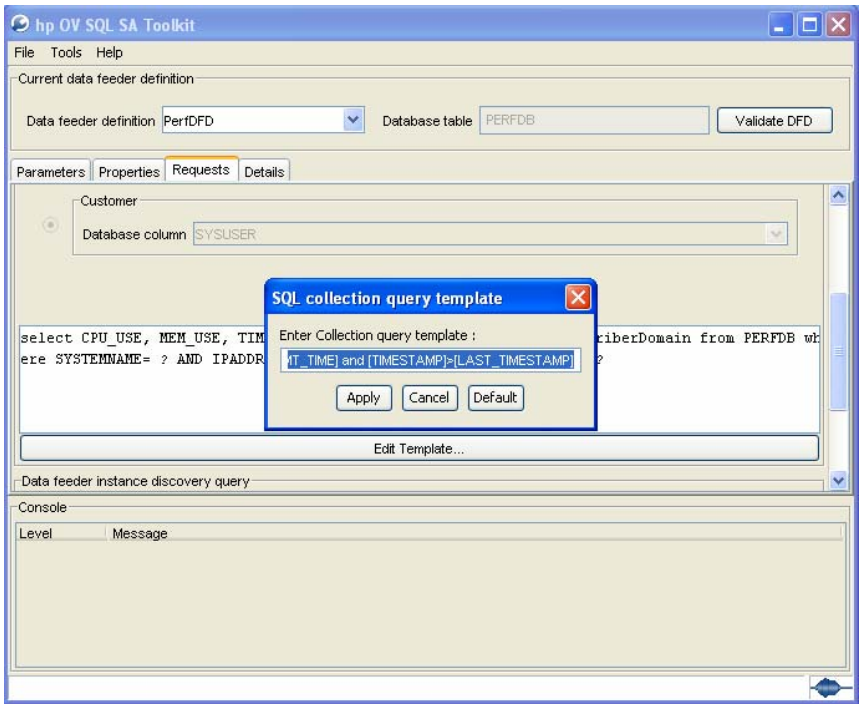
List of tables or view from which data will be selected (the table that has been associated to the Data Feeder definition phase). The keyword **[TABLES]**: will be replaced by table/view name associated to the Data Feeder.

For **WHERE** clause:

Restricts the rows selected to those for which the condition is TRUE. If you omit this clause, the database returns all rows from the tables or views in the FROM clause. The SQL Service Adapter supports the following keywords in this clause:

- **[MRP_PROPERTIES]** is replaced by DFI properties columns name = property value which are part of the MRP.
- [CURRENT_SYSTEM_TIME] current local time
- **[POLLING_PERIOD_BEFORE_CURRENT_GMT_TIME]** `current GMT time - Service Adapter Polling period value`
- **[LAST_POLL_TIME_BEFORE_CURRENT_GMT_TIME]** this is the last GMT time at which a given DFI data has been polled. At the first polling period this keyword is equal to the current time in GMT - Service Adapter Polling period value.
- **[LAST_TIMESTAMP]** this keyword can be used if a timestamp is defined in the database table. It represents the last timestamp value retrieved for a given DFI collection. When multiple rows are retrieved for a DFI in a single polling period, the greater timestamp is kept.

If the user wants to modify the default collection query for this Data Feeder, he edits the template (using **Edit template…** button), and manually define the SQL collection request.




After modification of the collection query, click on **Apply** button to accept changes or **Cancel**. The **Default** button retrieves the default collection query.

The Toolkit does not provide a test function. The user should copy/paste the collection query presented in the main windows into an SQL editor. Replace the "**?**" characters by real **TIMESTAMP** and **MRP PROPERTIES**. Then run the query to check that it returns the expected values.

## 3.8 Defining the DFI discovery request

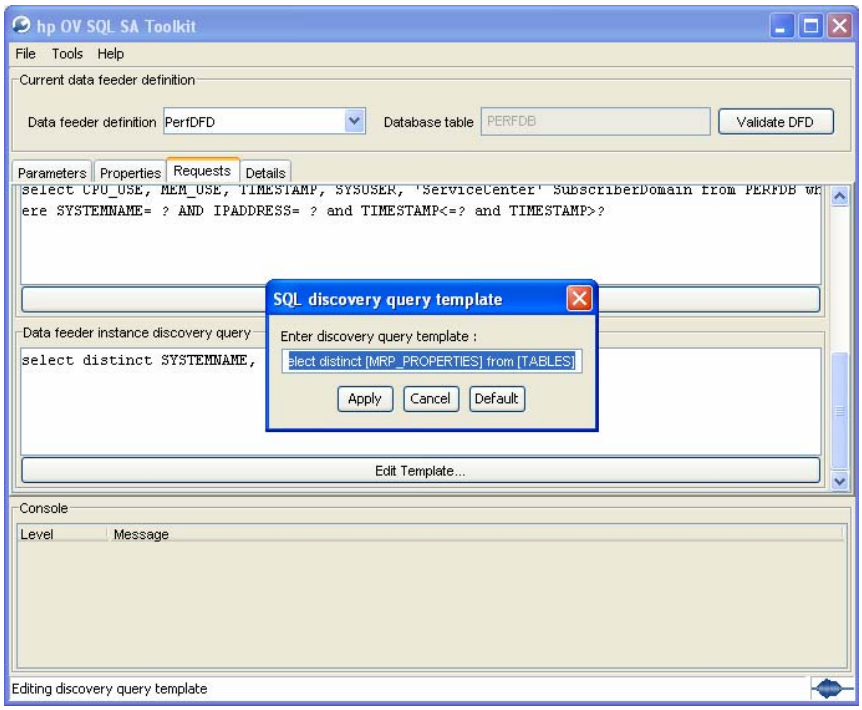
The DFI discovery request will be used by the SQL SA customization to create the Data Feeder Instances with their property values.

For the discovery request, a default request is also provided from the property mapping that has been defined in a previous customization step.

The default discovery query:

```
Select distinct [MRP_PROPERTIES] from [TABLES]
```

The user may decide to edit the default request and modify it if necessary using **Edit Template…** button, in the ***Data feeder instance discovery*** pane.

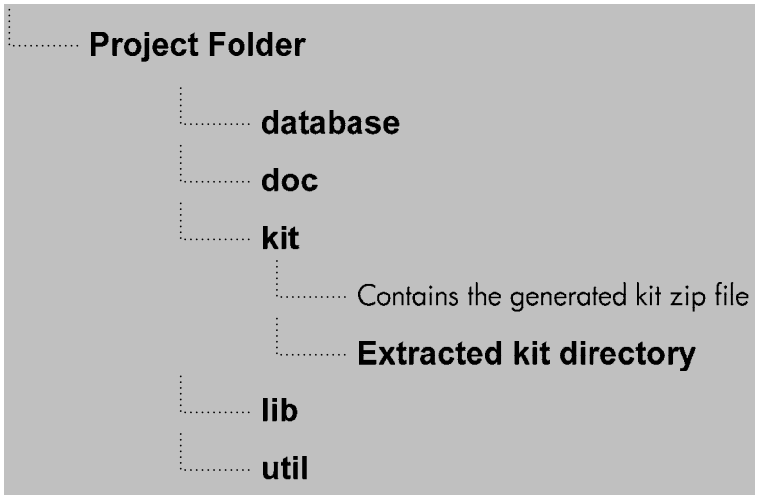



This request should return in the **SELECT** clause, all the DFD property values that will identify a DFI.

The toolkit does not provide a test function. The user should copy/paste the DFI discovery query into an SQL editor, run it, and check that the query returns the expected DFD property values.

## 3.9 Generating the Customized SA

As soon as the project has been defined (the first step of the customization), the following project directory layout is created:




If the user wants to include some specific documentation, scripts or utilities, the following directories should be augmented:

- **database** directory contains the SQL files that will be used to create the custom views within the database. The default mechanism will handle these SQL files, one by one, following the alphabetical order of their names.

Therefore, the **best practice** is to **name the files**: *01_my_global_stuff.sql*, *02_my_types.sql*, *03_my_first_view.sql*, *04_my_special_view.sql*, etc. Thus, the content of all the existing *.sql* files gets executed on the database server, in the right order, and therefore the associated types and views will be created. In case this convenient mechanism is not adequate, the user can use a different mechanism by adding, to this directory, scripts named ***temip_sc_create_db_view.sh*** (for Unix) and ***temip_sc_create_db_view.bat*** (for Windows). These alternate scripts have to keep these names, in order to allow the `temip_sc_configure -view` command to execute them, during the installation and configuration of the generated SQL SA custom. For details on the `temip_sc_configure` command, refer to paragraph *4.2 Configuring a customized SQL SA application*

- **doc** directory contains the SQL Service Adapter customization user documentation
- **lib** directory contains the JDBC drivers associated to the SQL database
- **util** directory contains any user specific utilities.

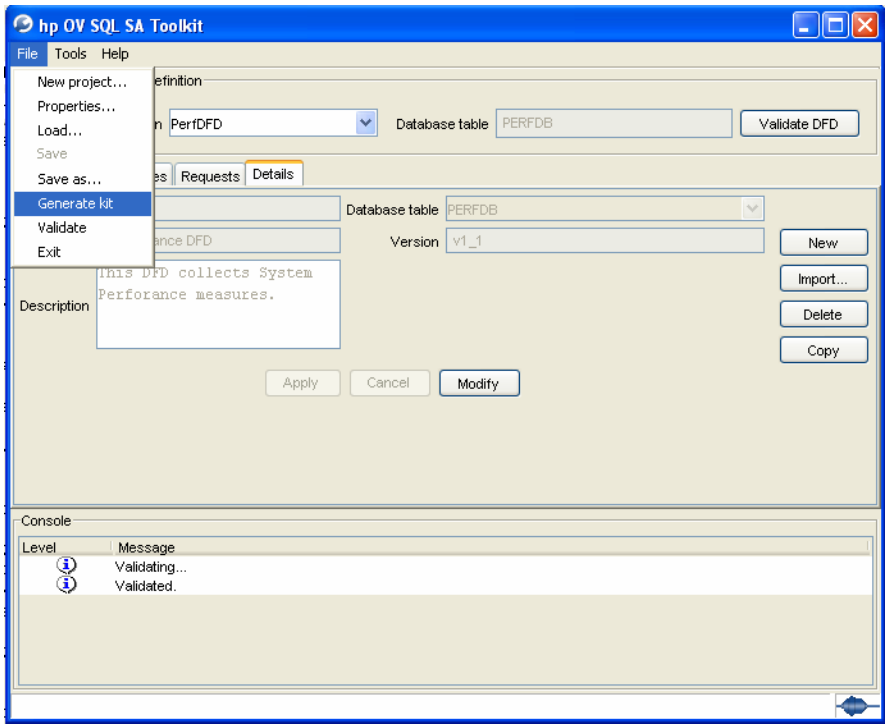
**Note**

All the JDBC drivers jar files necessary for running the SQL SA Custom have to be placed into the **lib** directory before generating the SQL SA Customization kit. The *Generate Kit* function will automatically include these drivers into the CLASSPATH of the SQL Service Adapter launcher.

Even if this JDBC CLASSPATH is updated for the SQL SA Customization launcher, this variable has to be part of the standard SQM environment variables on the system where the customization will run. So, at the customization installation, do not forget to update the **temip_sc_env.sh** file on Unix, or the **temip_sc_env.bat** file on Windows as described in chapter **Configuring the JDBC driver CLASSPATH**.

After the project directories have been filled, the customization kit can be validated. By selecting the menu **File** -> **Validate** the SQL SA Toolkit will validate that the project parameters and DFD mappings are correct before generating the kit file.

From the SQL Service Adapter Toolkit menu **File** -> **Generate kit**, the user can generate the zip file of the SQL SA Customization kit.



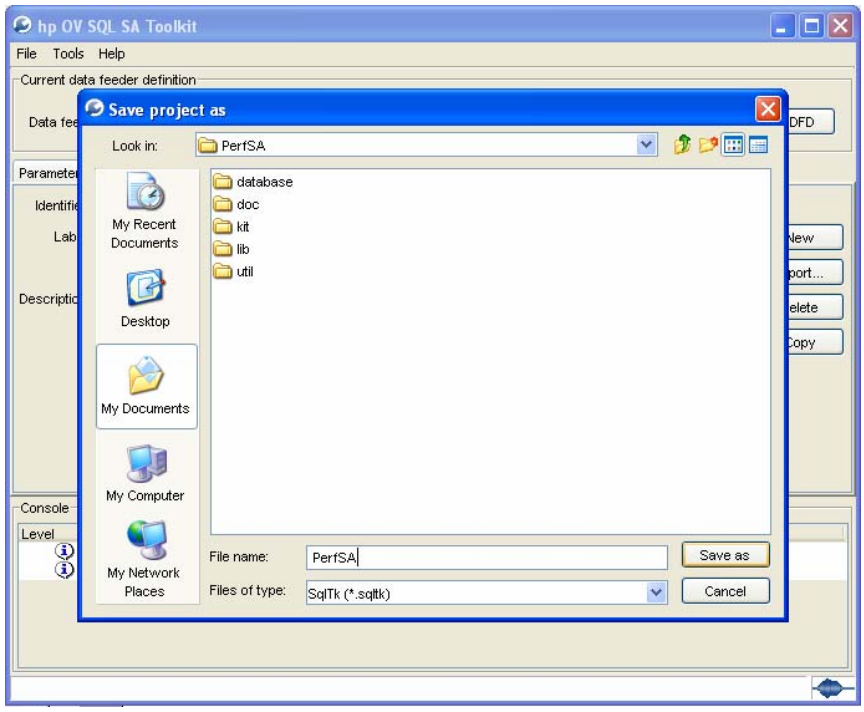

The kit zip file will be generated at the following location:

`<Project Directory>/kit/<KitName>.zip`

# 3.10 Saving the SQL SA Custom Project

The SQL SA Customization project can be saved, using the menu:
`File-> Save As…`

The user then selects the folder and provides the file name of the customization to be saved. At this file name will be added the SQL SA Toolkit **.sqltk** extension.



The by clicking on the button "**Save as"** the project will be saved.

## 3.11 Testing the customization

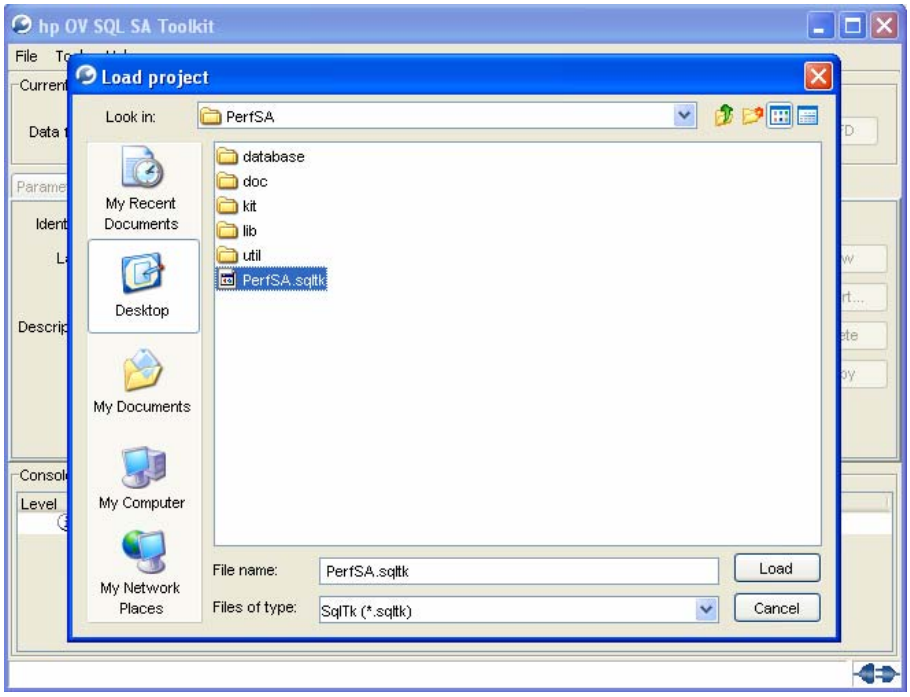
The SQL SA Toolkit does not provide a testing environment, but the user can at least copy/paste the collection and discovery queries to execute them on the targeted database in order to check the returned values.

In order to execute the queries on the targeted database, either use an SQL editor, or use the temip_sc_start_sqlexec tool. For details refer to 5.2.3 How to execute SQL statements (or SQL view creations) on the database?.

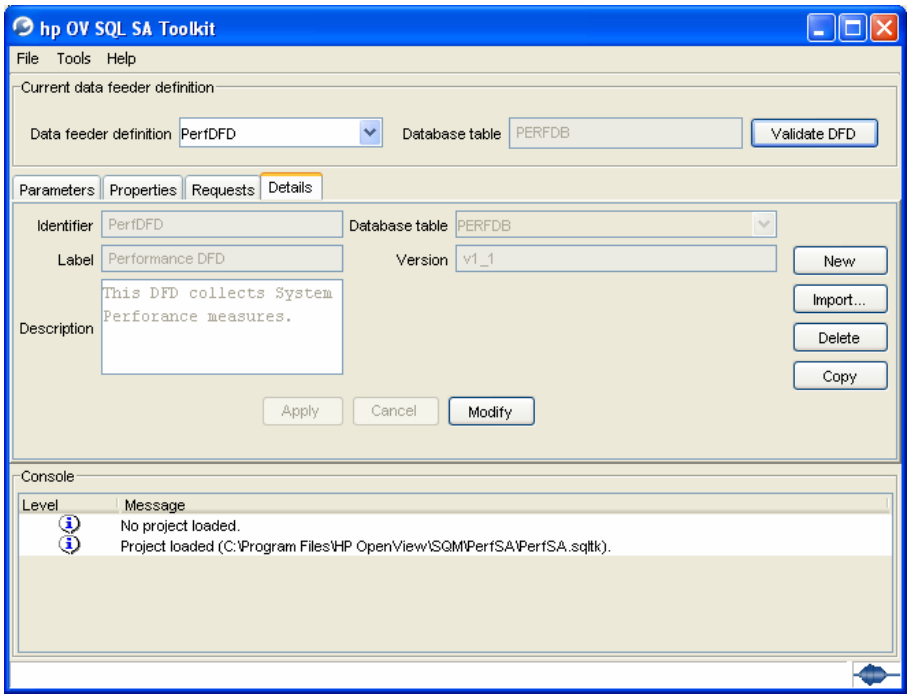
## 3.12 Modifying the customization

To modify an existing customization, the user can reload a previously saved SQL SA Toolkit project using the menu:
`File -> Load…`

Then select an existing SQL SA Custom project file.




After clicking on the **Load** button, the main SQL SA Toolkit window appears and the user can access the Data Feeders definition and the mapping information.




Modifying the existing mappings and queries, and adding or deleting DFDs is possible from main window tabs.

When all modifications are validated (with the **File** -> **Validate** menu) the user can re-generate the customization kit for testing.

# Chapter 4

# SQL SA customization installation and configuration

## 4.1 Installing a customized SQL SA

### 4.1.1 Software requirements

As for the SQL SA Toolkit, the SQL SA Customization kit requires that the following software:

- HP-UX V11.11 or Windows 2000 / Windows XP
- HP OpenView Service Quality Manager V1.3 (Kernel subset)
- HP OpenView SA Common V1.3 (**SQMSAGTWCOMMON**)
- HP OpenView SQL Service Adapter Runtime V1.3 (**SQMSASQL**)
- SQL Database JDBC driver

Prior to the SQL SA Customization installation, the SQM Kernel, The Service Adapter Common component (**SQMSAGTWCOMMON**), and the SQL SA Runtime V1.3 (**SQMSASQL**) will have been installed. Please refer to the previous SQL SA Toolkit installation chapter 2.3 for details.

### 4.1.2 Installing on HP-UX

On HP-UX, here are the steps to install the SQL SA Customization zip file. The Customization zip file has been previously generated by the Toolkit into **`<Project Directory>/kit/<KitName>.zip`**

- Connect as "**root**" user
- Set the TEMIP_SC_HOME environment variable to the SQM Root directory:

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

- Change directory to the SQM Root:

```
# cd $TEMIP_SC_HOME
```

- Unzip the SQL Service Adapter custom kit.

```
# unzip <Kit Full Path>
```

- Execute the installation script to set the correct access rights to the installed files:

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

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

where the <SACustomName> and <SAversion> are the SA name and Version
provided at the SQL SA Toolkit project creation.

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

# sh temip_sc_complete_install.sh
```

### 4.1.3 Installing on Windows

For an installation on Windows:

- Unzip the SQL Service Adapter custom kit in the **%TEMIP_SC_HOME%** directory.
- (the file has been previously generated by the Toolkit at the following location: **<Project Directory>\kit\<KitName>.zip**)

### 4.1.4 Configuring the SQM Kernel

The SQM Kernel needs to be setup to run a SQL SA Customization. To perform the setup, the following configurations may be considered:

1. The SQL SA customization 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 Server.

2. The SQL SA Customization 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:

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

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

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

3. The SQL SA customization is installed on Windows where the SQM Kernel has not been configured:

It is also necessary to retrieve the SLM Primary Server platform description file:

a. Retrieve the file ***$TEMIP_SC_VAR_HOME/setupconfig/platform_desc.cfg*** from the SQM SLM Primary Server, and copy on the SQL SA Customization Windows system in ***%TEMIP_SC_HOME%/tmp***

b. Open a command line and execute:

```
cd %TEMIP_SC_HOME%\setup\bin
temip_sc_setup -all -NI
```

# 4.2 Configuring a customized SQL SA application

#### General processing

**Important Note**

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

The setup and configuration of the SQL Service Adapter customization is done in 3 steps:

The application setup step, which declares the SQL SA application into the SQM Central Repository and create the SQL SA Customization application Data Tree into **TEMIP_SC_VAR_HOME**.

The application configuration step, which prompts the user for the connection characteristics, and load the connection and Data Feeder definitions into the SQM Central Repository.

The database view creation step, which executes the user defined SQL views creation files, delivered within the SQL SA customization directory: **database**. This SQL files has to be provided during the customization steps (refer to **Generating the Customized SA** chapter).

## 4.2.1 Configuring on HP-UX

The configuration tool is located in:

**$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin**

where the ***<SACustomName>*** and ***<SAversion>*** are the **SA name** and **Version** provided at the SQL SA Toolkit project creation.

### 4.2.1.1 Application setup and configuration steps

The following steps are to be performed to setup and configure the application on Unix:

#### Application creation

This phase consists in creating a SQL SA customization application on the SQM platform (on a specified director) and setup the SQM kernel if necessary.

Command:

Connect as "**root**" user.

Load the SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)

Perform the following commands:

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

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

```
name>

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

where <director name> is the director on which will be created the
application. (by default the director name is acquisition)
```

The application name has to be provided by the user.

Output:

This command will declare the SQL Service Adapter application to the SQM Central Repository and create the application Data Tree in:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<
SAversion>
```

As follows:

```
TEMIP_SC_VAR_HOME/Service Adapters/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/config/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/config/
SaSqlDiscoveryMtLogging.properties
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/config/
SaSqlDiscoveryTraceLogging.properties
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/config/
slmv120_acquisitionHP_perfSA.properties
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/discove
ry/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/discove
ry/filter/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/discove
ry/inventory/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/discove
ry/inventory/raw/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustoName_SAVersion/discover
y/inventory/filtered/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/discove
ry/repository/
TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/SACustomName_SAVersion/reposit
ory/
```

### Application configuration

The configuration consists in defining the SQL SA connection parameters and loading the connection and data feeder configuration into the SQM Central Repository.

Command:

Load the SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)

Perform the following command:

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

#temip_sc_configure.sh -configure <application name>

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

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

Output:

This command will prompt the user for SQL database connection information (login, password, database URL, JDBC driver) and load the SQL Service Adapter Data Feeders definitions and connector definition into the SQM Central Repository.

```
Configure the "perfSA" application ...

Please enter the database URL
[jdbc:oracle:thin:@hotel.vbe.cpqcorp.net:1521:spdm]:

Please enter the database username [spdm]:

Please enter the database password [********]:

Please enter the database JDBC Driver
[oracle.jdbc.driver.OracleDriver]:

Load the Connector in the Tibco Repository

Backup written at the following location:
/var/opt/OV/SQM/slmv120/ServiceAdapters/Sql/v1_2/PerfSA_v1_4/repos
itory/connectors_data.exp.2004_1_22_03_46_03

/var/opt/OV/SQM/slmv120/ServiceAdapters/Sql/v1 2/PerfSA v1 4/repos
itory/connectors_data.exp has been imported into the Repository

Load the Data Feeder Definitions in the SRM

load DFD: PerfDFD - v1 2
(../repository/NewDFDReq_PerfDFD.v1_2.xml) ... succeed.

Load the Data Feeder Definitions in the Tibco Repository

/var/opt/OV/SQM/slmv120/ServiceAdapters/Sql/v1 2/PerfSA v1 4

Backup written at the following location:
/var/opt/OV/SQM/slmv120/ServiceAdapters/Sql/v1_2/PerfSA_v1_4/repos
itory/PerfSA dfds data.exp.2004 1 22 03 46 18

../repository/PerfSA_dfds_data.exp has been imported into the
Repository

Configuration succeed.
```

### SQL view creation

If custom views are required, the user has to write the SQL files, used to create these custom views. During the customization of the SQL Service Adapter, the SQL files that have been placed into the SQL SA Toolkit project directory (refer to paragraph *3.9 Generating the Customized SA*) will be used by the following command to create the custom views within the database.

Command:

Load the SQM environment variables ($TEMIP_SC_VAR_HOME/temip_sc_env.sh)

Perform the following command:

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

#temip_sc_configure.sh –view <application name>

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

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

### 4.2.2 Configuring on Windows

The configuration tool is located in:

`%TEMIP_SC_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\bin`

where the `<SACustomName>` and `<SAversion>` are the **SA name** and **Version** provided at the SQL SA Toolkit project creation.

#### 4.2.2.1 Application setup and configuration steps

Here are the commands to be performed to setup and configure the application on Windows:

##### Application creation

This phase consists in creating a SQL SA customization application on the SQM platform, on a specified director.

Command:

Open a Command Line

Perform the following commands

```
# call "%TEMIP_SC_VAR_HOME%"\temip_sc_env.bat

# cd
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\bin

#temip_sc_configure -setup -dirName <director name> <application name>

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

where <director name> is the director on which will be created the
application (by default the director name is acquisition).
```

The application name has to be provided by the user.

Do not create two Service Adapters with the same SA application name on 2 different directors.

Output:

This command will declare the SQL Service Adapter application to the SQM Central Repository and create the application Data Tree in:

`%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>`

as follows:

```
TEMIP_SC_VAR_HOME\Service Adapters\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\config\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\config\
SaSqlDiscoveryMtLogging.properties
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\config\
SaSqlDiscoveryTraceLogging.properties
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\config\
slmv10_acquisitionWin_perfSA.properties
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\filter\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\inventory\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\inventory\raw\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\inventory\filtered\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\discove
ry\repository\
TEMIP_SC_VAR_HOME\ServiceAdapters\Sql\v1_2\SACustomName_SAVersion\reposit
ory\
```

### Application configuration

Open a Command Line

Perform the following commands

```
# call "%TEMIP_SC_VAR_HOME%"\temip_sc_env.bat

# cd
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\bin

#temip_sc_configure -configure <application name>

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

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

This command will prompt the user for SQL database connection information and load the SQL Service Adapter Data Feeders definitions and connector definition into the SQM Central Repository.

Output:

```
>>>temip_sc_configure -configure XPPerf
Configure the "XPPerf" application ...
Please enter the database URL
[jdbc:oracle:thin:hammam.vbe.cpqcorp.net:1521:spdm]:
Please enter the database username [spdm]:
Please enter the database password [********]:
Please enter the database JDBC Driver [oracle.jdbc.driver.OracleDriver]:
Load the Connector in the Tibco Repository
Backup written at the following location: C:\Program Files\HP
OpenView\SQM\slmv120\ServiceAdapters\Sql\v1_2\PerfSA_v1_4\repository\
connectors_data.exp.2004_1_22_15_25_11
C:\Program Files\HP
OpenView\SQM\slmv120\ServiceAdapters\Sql\v1_2\PerfSA_v1_4\repository\
connectors_data.exp has been imported into the Repository
Load the Data Feeder Definitions in the SRM
load DFD: PerfDFD - v1_2 (..\repository\NewDFDReq_PerfDFD.v1_2.xml) ... succeed.
Load the Data Feeder Definitions in the Tibco Repository
C:\Program Files\HP OpenView\SQM\slmv120\ServiceAdapters\Sql\v1_2\PerfSA_v1_4
Backup written at the following location: C:\Program Files\HP OpenView\SQM\slmv1
10\ServiceAdapters\Sql\v1_2\PerfSA_v1_4\repository\PerfSA_dfds_data.exp.2004_1_2
2_15_25_24
..\repository\PerfSA_dfds_data.exp has been imported into the Repository
Configuration succeed.
```

#### SQL view creation

If custom views are required, the user has to write the SQL files, used to create these custom views. During the customization of the SQL Service Adapter, the SQL files that have been placed into the SQL SA Toolkit project directory (refer to paragraph *3.9 Generating the Customized SA*) will be used by the following command to create the custom views within the database.

Command:

```
# call "%TEMIP_SC_VAR_HOME%"\temip_sc_env.bat

# cd
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\bin

# temip_sc_configure –view <application name>

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

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

### 4.2.3 Configuring the JDBC driver CLASSPATH

As for the SQL SA Toolkit, the variable **TEMIP_SC_JDBC_DRIVER_CLASSPATH** has to be configured.

If the user has placed the drivers in the SQL SA Customization project folder (refer to **Generating the Customized SA** chapter), the drivers should be available in the following directory:

#### On Windows:

```
%TEMIP_SC_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAv
ersion>\lib
```

- where the <***SACustomName***> and <***SAversion***> are the **SA name** and **Version** provided at the SQL SA Toolkit project creation

**On Unix:**

- $TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<***SACustomName***>_<***SAversion***>/lib
- where the <SA Custom Name> and <version> are the **SA name** and **Version** provided at the SQL SA Toolkit project creation
- If it is not the case, the user has to install the driver(s) on the system where the SQL SA Customization will run.
- In all cases, SQM environment variable file has to be updated as follows:

**On Windows:**

- Edit the file %TEMIP_SC_VAR_HOME%\temip_sc_env.bat
- Augment the file with:

  - set TEMIP_SC_JDBC_DRIVER_CLASSPATH=<Driver 1 FIllPath>;<Driver2 FullPath>; ….

**Note**

On Windows, the driver paths have to be separated by "**;**" characters and no "**"**" character should be included.

**On Unix:**

- Edit the file $TEMIP_SC_VAR_HOME/temip_sc_env.sh
- Augment the file with:

  - TEMIP_SC_JDBC_DRIVER_CLASSPATH=<Driver 1 FIllPath>:<Driver2 FullPath>: ….
  - export TEMIP_SC_JDBC_DRIVER_CLASSPATH

**Note**

On Unix, the driver paths have to be separated by "**:**" characters.

**Important Note**

**After the variable TEMIP_SC_JDBC_DRIVER_CLASSPATH has been updated, it is necessary to source the environment.**

**Restarting the SQM platform and Kernel is required only if at the SQL SA Customization kit generation, the drivers have not been included in the generated kit.**

### 4.2.4 Configuring the SQL SA Runtime license

The SQM utility to manage the licenses is ***temip_sc_license***.

To run a SQL SA Customization, it is necessary to configure the **SQM-SA-GENERATED_RT** license. There should be one **SQM-SA-GENERATED_RT** license per SQL SA Custom installed on the system.

At Kernel installation, a temporary **SQM-SA-GENERATED_RT** license is installed. This temporary license allows a 90 days trial period and is activated on the first license check (when the application is started the first time).

Once Autopass is installed and SQM Kernel configured (this has been done by the previous steps), perform the following commands to check the licenses.

#### On Windows:

Open a command line and execute:

```
> cd %TEMIP_SC_VAR_HOME%\bin
> temip_sc_license -check
```

#### On Unix:

Connect as "**root**" user

Load the SQM environment variables
($TEMIP_SC_VAR_HOME/temip_sc_env.sh)

Perform the following command

```
# temip_sc_license -check
```

When this temporary license key has expired, the SQL SA Customization runs in degraded mode. A permanent license is then required. Please refer to the *hp Openview SQM Administration Guide* where is explained how to retrieve SQM licenses and import them into Autopass (with ***temip_sc_license*** utility).

## 4.2.5 Advanced application configuration

This chapter describes how to setup the SQL Service Adapter 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.

### 4.2.5.1 Application, connection and DFD configuration variables

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

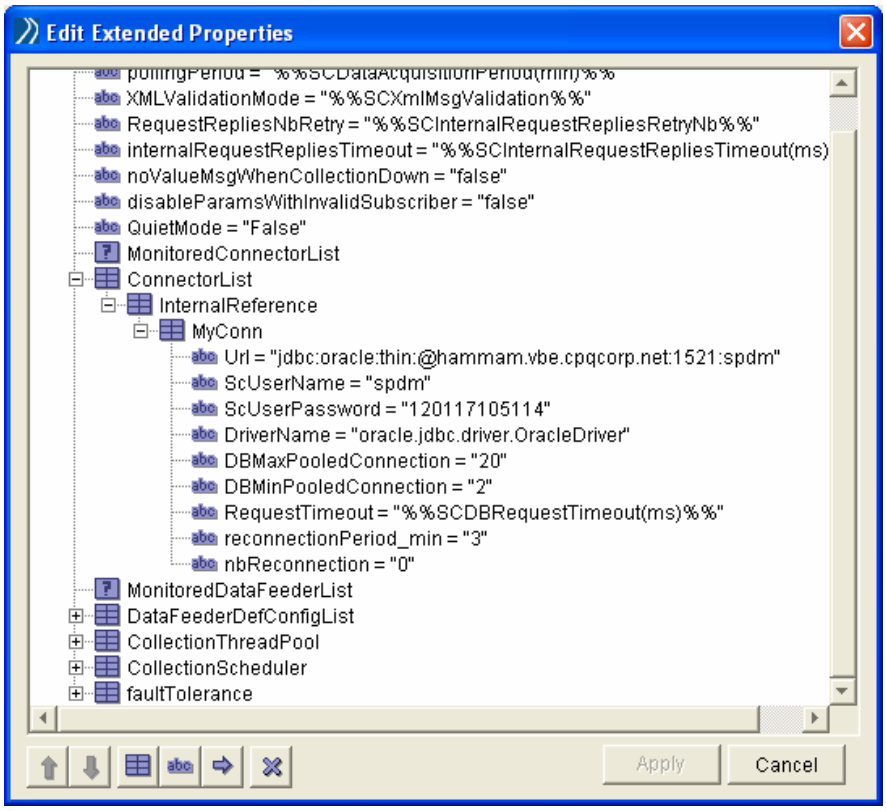
/ tibco / private / adapter / ServiceCenter / ServiceAdapters / <SACustomName>/<SA version>/<application name>_config

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 SQL SA behavior.

| Variable Name | Default | Description |
|---|---|---|
| pollingPeriod | %%SCData Acquisition Period(min) %%.<br>(Global variable) | This polling period applies to all the data collection by all of the data feeder instances managed by this service adapter application. Each time this period is reached, each data feeder instance gets the parameter values coming from the SQL database. This does not mean that all data collections are done at the same time, but that their frequency is equal to the pollingPeriod. This variable is applicable to the SQL Service Adapter. It is recommended to set it, depending the SQL database data refresh rate. It has to be greater to this database refresh period.<br>The minimum pollingPeriod is 0.5 which corresponds to 30 seconds. |
| MonitoredConnectorList | | The MonitoredConnectorList variable contains a list of the connector's logical names that are used by this service adapter instance. |
| ConnectorList | | The ConnectorList variable contains all the connection data. All the monitored connectors should be found inside the configuration list. |
| MonitoredDataFeederList | | The MonitoredDataFeederList variable contains a list of the data feeder definitions managed by this service adapter instance. Each data feeder definition is identified by its name and its version. |
| DataFeederDefConfigList | | The DataFeederDefConfigList variable contains all the configuration data of Data Feeder Definitions. All the monitored Data Feeder Definitions should be found inside the configuration list. |

SQL Database connection parameters can be configured by opening the SQM Central Repository and editing the same URL:

/ tibco / private / adapter / ServiceCenter / ServiceAdapters / <SACustomName> / SA<version>/ <application name>_config



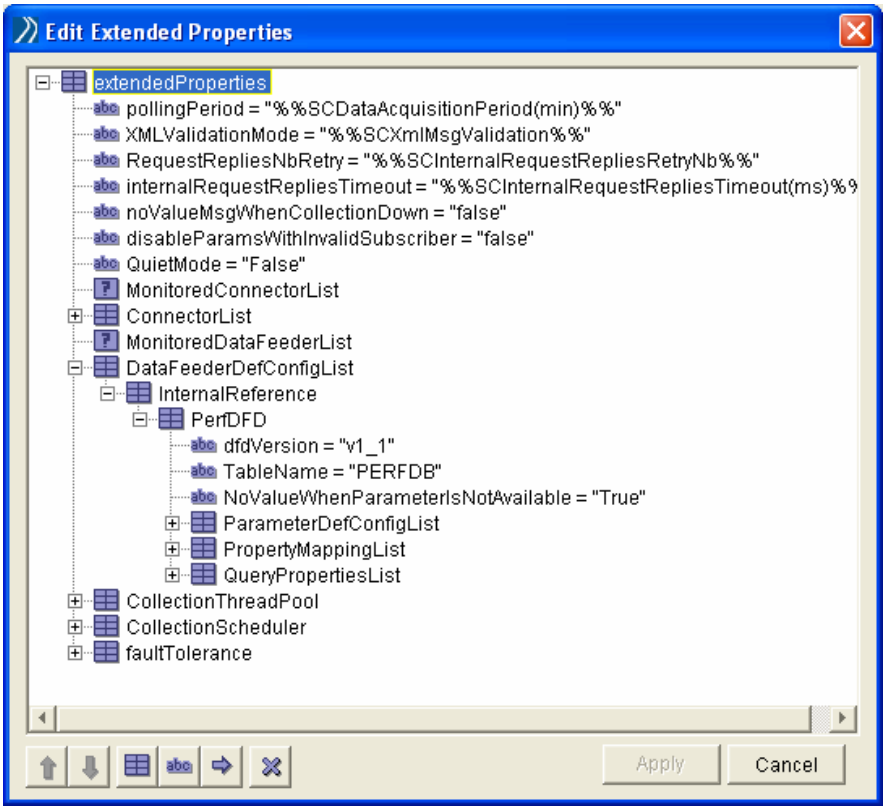

In the **ConnectorList** variable the SQL database connection is described, and the following parameters can be configured:

| Variable Name | Default | Description |
|---|---|---|
| reconnectionPeriod_min | 3 | When the connection is in error, a polling mechanism starts and attempts to reopen the connection. The reconnection tentative is done at every reconnectionPeriod (the value is specified in minutes). |
| nbReconnection | 0 | The nbReconnection variable represents the maximum number of reconnection tentatives. When the value is “0”, this means attempting to reconnect an infinite number of times. |

| | | |
|---|---|---|
| DbMaxPooledConnection | 20 | This variable represents the maximum of simultaneous connections opened to the database.<br>Note: the database must be configured to support such a number of connections. For instance on an Oracle database this maximum number of connections is configured by the variable named 'processes' in the file located in: $ORACLE_HOME/../../admin/<database name>/pfile/init<database name>.ora |
| DbMinPooledConnection | 2 | Minimum number of connections initialized by the SQL SA to the database. |
| RequestTimeout | %%SCDRequestTimeout(ms)%%.<br>(Global variable) | Request timeout for all requests towards Database. |

SQL SA Data Feeder definition parameters can be configured by opening the SQM Central Repository and editing the same URL:

/ tibco / private / adapter / ServiceCenter / ServiceAdapters / <SACustomName> / <SAversion>/ <application name>_config



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. |

#### 4.2.5.2 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)

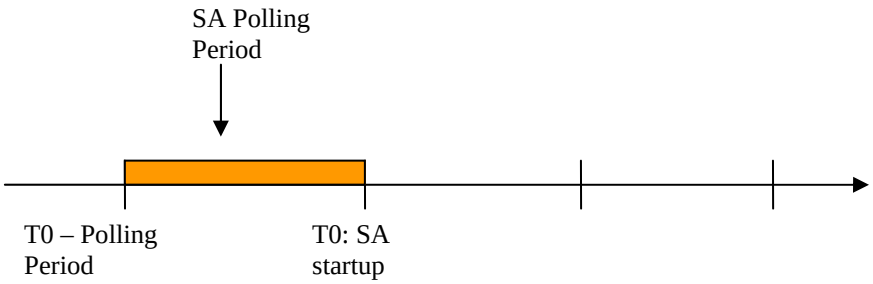
RequestRepliesNbRetry

internalRequestRepliesTimeout

#### 4.2.5.3 SQL Init collection query

At startup (or at restart), the current behavior of the SQL SA is to collect data available in the following time window:

**[ Current Time – Polling Period]** → **Current Time**




Therefore, if no data is available in the database on this period of time, no value is published by the SQL SA at startup or restart. It will be necessary to wait for the next polling period (if new value has been inserted into the database).

To provide initial parameter values to SQM at a startup or restart of the Service Adapter (and by the way, synchronize SQM with last values inserted into the database), the user may customize a specific collection query that will be executed when the application is started.

This query is called *the Initialization Query*. This query is called each time a Data Feeder Instance is unlocked. It means at Service Adapter Startup when the DFI is unlocked and if the DFI is locked when SA is running at the next unlock message the initial request will be executed for the DFI.

An example of default init query is defined as follows (for ORACLE):

```
SELECT [SUBSCRIBER], [SUBSCRIBER_DOMAIN],
[PARAMETERS], [TIMESTAMP] FROM (SELECT [SUBSCRIBER],
[SUBSCRIBER_DOMAIN] , [PARAMETERS], [TIMESTAMP] FROM
[TABLES] WHERE [MRP_PROPERTIES] AND
[TIMESTAMP]<=[CURRENT GMT_TIME] ORDER BY [TIMESTAMP]
DESC) WHERE rownum = 1
```

The Initialization query has to return the same parameters as the collection query, and in the same order. Except that it selects the row that has the most recent timestamp. The timestamp extracted by this query will be assigned to the keyword [LAST_TIMESTAMP] of the SQL SA. This keyword in then used by the collection query (if the user has not modified the default collection query) as explained in the chapter **Defining the collection request).**

The initialization query has to be customized in the following file.

**On Windows:**

```
"%TEMIP_SC_HOME%"\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\properties\TeSC<SACustomName>.properties
```

**On Unix:**

```
$TEMIP_SC_HOME\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\properties\TeSC<SACustomName>.properties
```

The default initialization query is applicable for all DFD.

#### Default Initialization query (example for ORACLE)

```
Sql.DefaultInitQuery.Pattern=SELECT [SUBSCRIBER],
[SUBSCRIBER DOMAIN], [PARAMETERS], [TIMESTAMP] FROM
(SELECT [SUBSCRIBER], [SUBSCRIBER_DOMAIN] ,
[PARAMETERS], [TIMESTAMP] FROM [TABLES] WHERE
[MRP PROPERTIES] AND [TIMESTAMP]<=[CURRENT GMT_TIME]
ORDER BY [TIMESTAMP] DESC) WHERE rownum = 1
```

This query has to be adapted to the chosen database type (ex: MS SQL, MySQL…).

But the user may provide a specific initialization query per DFD, and in this case the following syntax is applicable (in the SQL query property file):

#### Initialization query per DFD

```
Sql.InitQuery.Pattern.<DFD Name>.<DFD
Version>=<Customized SQL initialization template
request>
```

Example:

```
Sql.InitQuery.Pattern.MyDFD.v1_2 = SELECT
[SUBSCRIBER], [SUBSCRIBER_DOMAIN] , [PARAMETERS],
[TIMESTAMP] FROM (SELECT [SUBSCRIBER],
[SUBSCRIBER_DOMAIN] , [PARAMETERS], [TIMESTAMP] FROM
[TABLES] WHERE [MRP_PROPERTIES] AND
[TIMESTAMP]<=[CURRENT_GMT_TIME] ORDER BY [TIMESTAMP]
DESC) WHERE rownum = 1
```

#### SA behavior at startup and reconnection

If the default initialization request is defined and no specific DFD initialization request exists:

For the first collection, this default initialization request is executed.

For the other polling period the collection request is used.

If a DFD initialization request is defined:

For the first collection, this DFD initialization request is executed.

For the other polling period the collection request is used.

If no default or DFD initialization request are defined:

No initialization, but only collection requests are executed.

### 4.2.5.4 Time synchronization

Synchronizing the targeted SQL database (especially the SQL timestamp column value used for the data collection) with the SQM system where the SQL SA is running, is mandatory. Without this, the SQL SA application may never return the expected column data.

The user may choose to use NTP or other existing tool to align system clock.

## 4.3 Discovering and loading DFIs

The DFI discovery is an important feature provided by the SQL Service Adapter. It allows discovering and loading automatically the Data Feeder Instances that will be managed by the SQL SA Application.

### Discovery script

The discovery script is located in the following directory:

### On Windows:

`%TEMIP_SC_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\bin\temip_sc_discovery.bat`

where the ***<SA Custom Name>*** and ***<SAversion>*** are the **SA name** and **Version** provided at the SQL SA Toolkit project creation.

### On Unix:

`$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin/temip_sc_discovery.sh`

### General Processing

The discovery is done in 3 steps:

Raw discovery phase: that performs the discovery request defined by the customization, and retrieves all discovered DFIs 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 (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.

**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.

### Pre-requisite

The SQM SRM must be started before performing the Data Feeders discovery. To start it:

Connect as "**sqmadm**" user.

Load the SQM environment variables

(default: /var/opt/OV/SQM/slmv12/temip_sc_env.sh)

Perform the following commands:

```
# temip_sc_start_application -platform slmv12 -director slmonitoring -
application SRM
```

### 4.3.1 Raw discovery phase

This initial phase will load the discovery request definition file and perform the discovery query to retrieve all the SQL SA Data Feeder Instance definitions.

#### Input

To perform the Raw discovery, the SQL SA will load the Discovery query that has been defined by the customization (refer to **Defining the DFI discovery** request chapter).

This request has been generated in the customization kit, and is available at:

#### On Unix:

$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<***SACustomName***>_<***SAversion***>/properties/TeSCSqlDiscovery.xml

#### On Windows:

%TEMIP_SC_HOME%\ServiceAdapters\Sql\v1_2\<***SACustomName***>_<***SAversion***>\properties\TeSCSqlDiscovery.xml

**Note**

It is possible to specify a different discovery file by adding the option –cfg to the discovery tool:

temip_sc_discovery **–cfg** <discovery file>

Please refer **to Appendix C** for an example of discovery request file.

#### Command

The discovery request has to be performed as follows:

#### On Unix:

Connect as "**sqmadm**" user.

Load the SQM environment variables

(default: /var/opt/OV/SQM/slmv12/temip_sc_env.sh)

Perform the following commands:

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

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


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

     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_2\<SACustomName>_<SAversion>\bin

#temip_sc_discovery -platform <platform name> -director <director name> -
application <application name> -discover


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

     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 raw discovery phase output will generate the following files.

The discovered DFI inventory file, located in:

### On Unix:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/discovery/inventory/raw/***<platform name>_<director name>_<application name>***.xml

### On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2***<SACustomName>_<SAVersion>***\discovery\inventory\raw***<platform name>_<director name>_<application name>***.xml

An example of inventory file is provided in **Appendix D**.

The DFI declaration and deletion XML files, located in:

### On Unix:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/discovery/repository/***DeclareDFIReq_<DFDName>.<DFDversion>.<DFIID>***.xml

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/discovery/repository/***DelDFIReq_<DFDName>.<DFDversion>.<DFIID>***.xml

#### On Windows:

```
%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_
<SAVersion>\discovery\repository\DeclareDFIReq_<DFDName>.<DF
Dversion>.<DFIID>.xml
```

```
%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_
<SAVersion>\discovery\repository\DelDFIReq_<DFDName>.<DFDver
sion>.<DFIID>.xml
```

### 4.3.2 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_2/<SACustomName>_<
SAversion>/discovery/filter/<platform name>_<director
name>_<application name>_filter.sh
```

#### On Windows:

```
%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2\<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 E.

The raw DFI inventory file is located at:

#### On Unix:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<
SAVersion>/discovery/inventory/raw/<platform name>_<director
name>_<application name>.xml
```

### On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2***<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/slmv12/temip_sc_env.sh)

Perform the following commands

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<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_2\<SACustomName>_<SAversion>\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_2\<SACustomName>_<SAversion>\discovery\inventory\filtered\<platform name>_<director name>_<application name>.xml`

### 4.3.3 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_2/<***SACustomName***>_<***SA version***>/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_2***<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_2/***<SACustomName>_<SAversion>***/discovery/repository/***<platform name>_<director name>_<application name>***_discovery_reference.xml

### On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2***<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/slmv12/temip_sc_env.sh)

Perform the following commands

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<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_2\<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_2\<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_2\<SACustomName>_<SAversion>\discovery\repository\<platform name>_<director name>_<application name>_discovery_cmds.bat
```

### 4.3.4 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

#### On Unix:

Connect as "**sqmadm**" user.

Load the SQM environment variables

(default: /var/opt/OV/SQM/slmv12/temip_sc_env.sh)

Perform the following commands

```
# cd
$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/bin

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


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

     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_2\<SACustomName>_<SAversion>\bin

#temip_sc_discovery -platform <platform name> -director <director name> -
application <application name> -all


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

     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 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**)

## 4.3.5 Scheduling the DFI discovery

It is recommended to encapsulate all the previously DFI discovery commands into specific scripts that can run in a crontab (for Unix) or as a scheduled task (on Windows).

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

## 4.4 Starting/Stopping customized SQL SA

Starting and stopping an SQL Service Adapter 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:

### On Unix:

Connect as “sqmadm” user

Load the SQM environment variables

### On Windows:

Open a command line window

Load the SQM environment variables:

```
cd "%TEMIP_SC_VAR_HOME%

                call temip_sc_env.bat
```

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_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.
```

## 4.5 Upgrading a customized SQL SA

When a defined and deployed SQL SA Customization requires some enhancements (adding new DFDs, adding new DFD parameters), it is recommended to perform the customization upgrade using the following steps:

1. Open the Customization Project with the SQL SA Toolkit, and modify the project properties:
    a. **Project Folder**
    b. **SA Version (ex: v1_2)**
    c. **Kit Name (including the new version)**
2. Save your new customization project into another project name (using `File -> Save as` option)
3. Add DFDs (with their associated mapping) or Modify existing DFDs:
    1. Upgrade the DFD version (for existing DFDs) ex: **v1_2** instead of **v1_0**
    2. Add parameters and mapping definitions to the existing DFDs
    3. Add new DFDs and parameters mapping
4. Generate the SQL SA Customization for this new version.
5. Install the new SQL SA Version on the targeted host:
   The installation of this new version can be done on the same host as the previous customization: two versions of the same SQL SA can be installed and run on the same host.
6. Configure a new application for this new SQL SA version.
7. Discover the new application DFIs and load them into SQM.

**Note**

Prior the to these steps, the user will have upgraded its SQM Service Model to support the New DFDs (with their new versions) and have created the appropriate Service Component instances which will be associated to the new discovered Data Feeder instances

8. Now that the DFIs are loaded into SQM, the binding to the Service Component Instances can be done using the SQM SL Administration User Interface. The collection on the new Data Feeder Instances can then be started.

**Note**

When the old SQL SA version will become obsolete on the platform, its associated application/DFD/DFIs will be deleted using the following commands:

#temip_sc_delete_application

#temip_sc_delete_dfd

The detail of these commands is described in the *SQM Administration Guide*.

# 4.6 Deployment

## 4.6.1 Application distribution

As the SQL SA will need to be configured to connect to an SQL database, 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 SQL SA Customizations can run on the same host.

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

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

Technology driven. You use one director for each SQL SA Customization, meaning that all the applications of the same SQL SA customization share the same director.

OS driven. You use one director for each OS, so that, for example, all the SQL SA applications on Windows belong to Windows director and all the SQL SA 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 SQL SA applications collecting on databases located in Paris belong to the “Paris” director.

To group SQL SA 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.

### 4.6.2 Load balancing

Even if the number of applications is not limited on a SQM host, and the number of DFIs supported by a SQL SA 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 SQL SA application

Number of applications running on a single system

The performance of the targeted database (concurrent access to the tables and views)

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

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

Per DFD

Per DFI property (MRP)

Per customer

**Note**

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

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

The performance of the SQL SA Application can be tuned using the following parameters:

pollingPeriod

DbMaxPooledConnection (tuning this variable depends on the database type which support or not the connection pool)

DbMinPooledConnection (tuning this variable depends on the database type which support or not the connection pool)

Refer to the **Advanced application configuration** chapter for more information about these variables and on the next chapter (Performance tuning).

- About the database tuning, the DB administrator will make sure that:

The access to the database tables is optimized through a good partitioning of the information into multiple views (split the data per customer, geography, technology and map this partitioning to the DFD definition and DFI repartition).

The view of database table will provide the best performance for the SQL SA collection query.

#### 4.6.2.1 Examples of load balancing

The load balancing can be done thank to the discovery tools, by dispatching DFIs on multiple SQL SA applications. A scenario is explained here:

- There is a single Service Adapter Application managing a set of DFIs. It has been decided to create a second Service Adapter Application, to dispatch DFIs equitably of each application.

To dispatch the DFIs on both Service Adapter Applications perform the following actions:

1. It is supposed that a first Service Adapter Application is already running and the DFI discovery phases have been performed on this application
2. Create a new Service Adapter Applications (see chapter 4.2.2.1 for more information):
   # temip_sc_configure.sh –setup <application Name> -dirName <director name>
3. Configure the newly created application:
   # temip_sc_configure.sh –configure <application Name> -dirName <director name>
4. Perform the DFI discovery phase
   # temip_sc_discovery –application <application name> -director <director name> -platform <platform name> -discover
5. Filter manually (or use a script) discovered DFIs by:
   a. Copy the input raw inventory file to the filtered inventory file.

```
# cp
$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SA
CustomName>_<SAversion>/discovery/raw/<platform
>_<director>_<application>.xml
$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SA
CustomName>_<SAversion>/discovery/filtered/<pla
tform>_<director>_<application>.xml
```

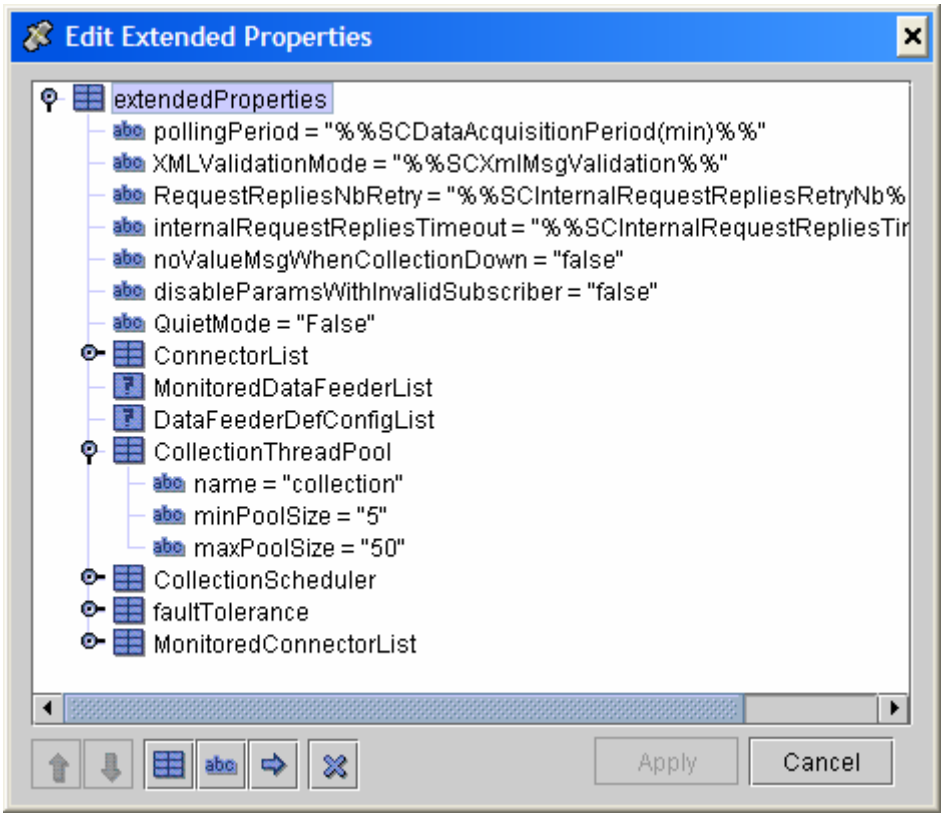
b. Edit the DFI filtered file to remove 50% of the discovered DFI (the other 50% are already managed by the initial SA application)

6. Load DFIs in the SRM
   # temip_sc_discovery –application <application name> -director <director name> -platform <platform name> -load
   DFIs initially assigned to the first SA application are now assigned to the second SA application
7. Stop/Start the first SA application
8. Start the second SA application

### 4.6.3 Performance tuning

As previously explained, the SQL SA performance depends on the following parameters:

- Number of DFIs
- Remote or local database
- Indexed SQL Table or not
- Number of connection allowed on the SQL database (SQL SA parameter DbMaxPooledConnection)
- Polling period
- Size of the collection pool. This parameter determines the maximum number of collection running in parallel. It can be configured in the Tibco repository by the Collection variable named ‘maxPoolSize’



The main important performance parameters are the number of DFIs to pool and the elapsed time to perform the SQL Request. The Service Adapter performance tuning will consist in increasing the following parameters according to the number of DFIs to pool:

- The number of connections allowed on the SQL database (SQL SA parameter DbMaxPooledConnection). Increase this parameter when DFIs number is greater than 1500.
  The following other parameters must be greater than DbMaxPooledConnection:
- Number of connections supported by the database: this parameter must be configured on the Oracle database. It is configurable by setting the variable named ‘processes’ in the file located in:
  $ORACLE_HOME/../../admin/<database name>/pfile/init<database name>.ora

  maxPoolSize of the collection pool which determines the maximum number of collections running in parallel. It can be configured in the Tibco repository by the Collection variable named ‘maxPoolSize’: maxPoolSize= DbMaxPooledConnection+20

  The collection polling period must be also increased according to the number of DFIs to poll. This parameter is determined by the variable named ‘pollingPeriod’ in the Tibco repository (see chapter Application, connection and DFD configuration variables)

# Chapter 5

# Advanced customization

## 5.1 Adding synthetics parameters

The SQL SA Toolkit allows defining synthetics parameters to a DFD. A synthetic parameter is a DFD parameter which is not mapped onto the SQL Table column but which is computed from a dedicated request.

For instance:

We need to add a parameter representing the number of rows retrieved at each request execution (SA polling period):

```
SELECT CPU_USE, COUNTER
FROM [TABLES],
        (SELECT COUNT(*) as COUNTER FROM [TABLES] where
        [MRP_PROPERTIES] and
        [TIMESTAMP]<=[CURRENT_GMT_TIME] and
        [TIMESTAMP]>[LAST_TIMESTAMP])
WHERE [MRP_PROPERTIES] and
        [TIMESTAMP]<=[CURRENT_GMT_TIME] and
        [TIMESTAMP]>[LAST_TIMESTAMP])
```

To define such parameter, when creating a new DFD parameter, instead of selecting a SQL column, enter a SQL column alias (the field is editable). This alias is computed by a sub-query.

**Note**

The timestamp field in the Request configuration windows is also editable

In the previous example, to define the COUNTER parameter:

Click on the '**Parameters**' tab in the toolkit main windows, the parameter mapping window will appear

Click on **New** button to create and edit the new 'COUNTER' parameter.

Fill in the ***Selected parameter details*** pane, for the field named '*Database column*', instead of selecting a SQL database column type the SQL alias name, here 'COUNTER'.

Click on the '**Requests'** tab in the toolkit main windows, the request configuration window will appear

Edit the '**Collection Query'** request by clicking on the 'Edit template …' button and enter the SQL query computing the COUNTER value:

```
SELECT [PARAMETERS], [TIMESTAMP]
FROM [TABLES],
        (SELECT COUNT(*) AS COUNTER
        FROM [TABLES]
```

```
        WHERE [MRP_PROPERTIES] AND
            [TIMESTAMP]<=[CURRENT_GMT_TIME] AND
            [TIMESTAMP]>[LAST_TIMESTAMP])
WHERE [MRP_PROPERTIES] AND
        [TIMESTAMP]<=[CURRENT_GMT_TIME] AND
        [TIMESTAMP]>[LAST_TIMESTAMP]
```

The instantiated SQL query will look like:

```
SELECT CPU_USE, COUNTER,TS
FROM PERFDB,
            (SELECT COUNT(*) AS COUNTER
            FROM PERFDB
            WHERE SYSTEMNAME= ? AND TS<=? AND TS>?)
WHERE SYSTEMNAME= ? AND TS<=? AND TS>?
```

# 5.2 Creating SQL views

## 5.2.1 Why creating views?

If the database has the possibility of managing views, the following remarks will be helpful in deciding to employ them or not.

- A view can be used to simplify the vision of a complex database for a particular user or application. The view may contain:

Only a summary of the table, rather than the full detail, using aggregation (group by).

Joined tables, combining tables that have been split due to normalization.

- A view can be used to give a name (and therefore a convenient way to use) a frequently used sub-query.
- A view can be used to make conversion of units (and therefore a convenient way to use).
- A view can be used for security. A user can be given access to a view, even if the user cannot access the underlying table. A user can be provided with an abridged or summarized view, while keeping certain data hidden.
- A view can be made read-only.

## 5.2.2 Creating views for SQL SA

The SQL SA Toolkit does not provide an automatic way to create views. Therefore the user has to create them and to test them.

The SQL Service Adapter has been designed to access database tables or views that map with the definition of the DFDs that it exposes to SQM. As explained in previous chapters, a DFD is associated to an SQL table that has the following column definition:

| | DFD Parameters | | | MRP properties | | | Subscriber | DFD Properties | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Timestamp** | Param1 | | ParamN | MRP1 | | MRPN | SubCol | Prop1 | | PropN |

If:

- The DFD parameters are to be collected from several tables,
- The DFD associated table does not have the previous characteristics,
- The DFD associated table contains columns with unsupported SQL data types,
- Some raw data columns request specific calculation to map to DFD parameters.

In this case, the creation of a view or the usage of SQL scripts is necessary.

#### Example

Please find below some useful example of view creation scripts.

The following view creates two columns NbOderIn and NbOrderOk which values are calculated from a number of rows of other tables.

```
create or replace view MNPLoading as
(select distinct o.COD_OLO as Operator,
(SELECT COUNT(*) FROM richieste_donor2 WHERE COD_DONATING = o.COD_OLO  AND
tipo_msg='01' AND num_file IN
(SELECT NUM_FILE FROM nome_file WHERE TRUNC(data_elab)=TRUNC(sysdate -1)
AND tipo='OLO')) as NbOrderIn,
(SELECT COUNT(*) FROM richieste_donor2 WHERE COD_DONATING = o.COD_OLO  AND
tipo_msg='01' AND num_file IN
(SELECT num_file FROM nome_file WHERE TRUNC(data_elab)=TRUNC(SYSDATE -1)
AND tipo='OLO')
AND num_file_olo_presa IS NOT NULL) as NbOrderOk,
(select to_char(sysdate - 1/24,'YYYY-MM-DD HH24:MI:SS') from dual) as GMT
from operator o);
```

### 5.2.3 How to execute SQL statements (or SQL view creations) on the database?

#### On Unix:

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

   (default: /var/opt/OV/SQM/slmv12/temip_sc_env.sh)
3. Perform the following commands

```
# cd $TEMIP_SC_HOME/bin

# temip_sc_start_sqlexec.sh -jdbcclass <jdbc driver class name> -url
<jdbc url for connection> -user <session user name> -passwd <session
user password> -requestFiles <file1 containing sql request> <file2>...


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

     where:

        the <jdbc driver class name> is the full qualified JDBC driver
```

entry class name provided by the database driver vendor.

the ***<jdbc url for connection>*** is the URL that allow the JDBC driver to uniquely identify the server and the database to which it should connect. The syntax of this URL is vendor specific.

the ***<session user name>*** is the user name to be used to connect to the database. This user has to have the adequate rights (role) to execute the request.

the ***<session user password>*** is the user password to be used for the specified user name, in order to connect to the database.

the ***<file1 containing sql request> <file2>...*** is the list of SQL files, whose content should be executed on the database. **CAUTION**: refer to the latter Note.

### On Windows:

1. Open a Command line window:

```
#cd "%TEMIP_SC_HOME%"\bin

#temip_sc_start_sqlexec -jdbcclass <jdbc driver class name> -url <jdbc
url for connection> -user <session user name> -passwd <session user
password> -requestFiles <file1 containing sql request> <file2>...


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

where:

the ***<jdbc driver class name>*** is the full qualified JDBC driver entry class name provided by the database driver vendor.

the ***<jdbc url for connection>*** is the URL that allow the JDBC driver to uniquely identify the server and the database to which it should connect. The syntax of this URL is vendor specific.

the ***<session user name>*** is the user name to be used to connect to the database. This user has to have the adequate rights (role) to execute the request.

the ***<session user password>*** is the user password to be used for the specified user name, in order to connect to the database.

the ***<file1 containing sql request> <file2>...*** is the list of SQL files, whose content should be executed on the database. **CAUTION**: refer to the below Note.

The SQL files have to contain only one statement at a time. Thus, for example, the SQL statements:

```
create or replace view myView1 as
((select distinct o.COD_OLO as Operator from operator o),
 (select to_char(sysdate - 1/24,'YYYY-MM-DD HH24:MI:SS') from dual) as
GMT);
create or replace view myView2 as
((select count(*)as OperatorCount from operator o),
 (select to_char(sysdate - 1/24,'YYYY-MM-DD HH24:MI:SS') from dual) as
GMT);
```

have to be split, and placed into two different files:

- myView1.sql

```
create or replace view myView1 as
((select distinct o.COD_OLO as Operator from operator o),
 (select to_char(sysdate - 1/24,'YYYY-MM-DD HH24:MI:SS') from dual) as
GMT)
```

- myView2.sql

```
create or replace view myView2 as
((select count(*)as OperatorCount from operator o),
 (select to_char(sysdate - 1/24,'YYYY-MM-DD HH24:MI:SS') from dual) as
GMT)
```

**Caution**

Remove the semi-colon characters from the end of the SQL statements, indeed only PL/SQL statements should be ended by a semi-colon character

# 5.3 Executing an Oracle PL/SQL function in a SQL SA request

## 5.3.1 Why performing an Oracle PL/SQL function?

In most cases the SQL request performed by the SQL SA look like:

**SELECT** [PARAMETERS], [TIMESTAMP] **FROM** [TABLE] **WHERE** [MRP_PROPERTIES] AND [TIMESTAMP]<=[CURRENT_GMT_TIME] AND [TIMESTAMP]>[LAST_TIMESTAMP]

When this type of request is not sufficient for our needs, by executing an Oracle PL/SQL request all limitations can be raised. In could help for:

- Performing a second SQL request which will for instance update the retrieved database rows
- Performing advanced synchronization requests. For instance marking the retrieved rows
- Any specific and complex requirement

A PL/SQL call function can be performed by executing the following kind of request:

**SELECT** [PARAMETERS], [TIMESTAMP] **FROM** *FUNCTION_NAME*([MRP_PROPERTIES],[CURRENT_GMT_TIME],[LAST_TIMESTAMP])

**Note**

A PL/SQL function can only be performed in the FROM clause of the SQL request. In this FROM clause the SQL SA will replace the keyword [MRP_PROPERTIES] by each properties part of MRP separated by the character ‘,’.

This rule is also applicable on the SQL Initial query.

## 5.3.2 How performing an Oracle PL/SQL function?

To call an Oracle PL/SQL function instead of perform a standard SQL request the recommended method is:

1. Define your PL/SQL function in a dedicated PL/SQL package. The function **must** return a SQL TABLE. It is recommended to:
2. Define a record SQL type. This record represents the result of the selection of a single table’s row. For instance if we expect to retrieve parameters ‘Param1’, ‘Param2’ and a timestamp for each DFI the record must define

these parameters and the timestamp. Put the definition of this data type in a SQL file located in the project directory named 'database'. This SQL file will be executed on the SQL Server during the view creation step of the Service Adapter configuration phase (temip_sc_configure –view ….) (Refer to the paragraph Sql View Creation in chapter Application setup and configuration steps). For instance, in a file named sql_01_record.sql:

```
CREATE OR REPLACE TYPE MY_RECORD AS OBJECT (
             CPU_USE Float,
             MEM_USE Float,
             DISK_USE Float,
             TS DATE);
```

3. Define a SQL Table based on this record. This is the SQL table returned by the PL/SQL function. As for the record definition, put this table definition in a SQL file. For instance, in a file named sql_02_table.sql:

```
CREATE OR REPLACE TYPE MY_TABLE IS TABLE OF MY_RECORD;
```

4. Declare a PL/SQL package for defining the PL/SQL function. This function must have in arguments which are the DFD MRP properties and least a timestamp. As for the record definition, put this declaration in a SQL file. For instance, in a file named sql_03_declare.sql:

```
CREATE OR REPLACE package MY_SA_PKG as
TYPE REF_CURSOR IS REF CURSOR;
FUNCTION retrieve_dfi_rows ( systemname_arg IN VARCHAR2,
                             current_gmt_time_arg IN DATE) RETURN
MY_TABLE;
END MY_SA_PKG;
```

5. Define the body of the PL/SQL function. As for the record definition, put this declaration in a SQL file. For instance, in a file named sql_04_body.sql. In the following example, we want to mark all retrieved SQL rows to resynchronise SQM with the 3PP if the SA is stopped. sql:

```
CREATE OR REPLACE package body  MY_SA_PKG as
       FUNCTION retrieve_dfi_rows ( systemname_arg IN VARCHAR2,
                                    current_gmt_time_arg IN DATE)
             RETURN  MY_TABLE IS  pragma autonomous_transaction;
             -- the pragma autonomous_transaction allows performing an
             -- update in the function
             ccur REF_CURSOR;
             retrieved_rows_tab MY_TABLE;
             BEGIN
             -- Perform the SQL Request to retrieve DFI rows on the given
             -- MRP property

             OPEN ccur FOR SELECT CAST (MULTISET(SELECT CPU_USE,MEM_USE,
             DISK_USE, TS FROM PERFDB WHERE SYSTEMNAME= systemname_arg
             AND TS <= current_gmt_time AND already_retrieved=0) AS
             MY_TABLE ) FROM DUAL;
             FETCH ccur INTO retrieved_rows_tab;
             CLOSE ccur;
```

```
-- Post execution: mark the retrieved Columns
UPDATE AVANTEL_ALARMS SET already_retrieved = 1 WHERE
SYSTEMNAME= systemname_arg AND TS <= current_gmt_time AND
already_retrieved=0;


COMMIT;


RETURN retrieved_rows_tab;
END;


END  MY_SA_PKG;
```

6. Customize your Service Adapter thank to the SQL SA Toolkit UI:

a) Instead of specifying a database table name in the Details tab of the main window, enter the call to the PL/SQL function (this call must be cast). You must enter in the database field:

```
TABLE(CAST(SA_PKG.RETRIEVE_DFI_ROWS([MRP_PROPERTIES],[CURREN
T_GMT_TIME]) AS MY_TABLE))
```

b) Define the DFD parameters and properties. As no table has been specified, it is impossible to select a database column. You must enter 'column' names manually. In fact here it is necessary to enter the defined SQL record field's name.

c) Define the Collection request. As the filtering is performed by the PL/SQL function the WHERE clause of the SQL request is no longer useful. The collection request is:

```
SELECT [PARAMETERS],[TIMESTAMP] FROM [TABLE]
The Sql SA will instantiate this template request as:
SELECT CPU_USE, MEM_USE, DISK_USE, TS FROM
TABLE(CAST(SA_PKG.RETRIEVE_DFI_ROWS(SYSTEM_NAME,<CURRENT_GMT
_TIME>) AS MY_TABLE))
```

d) Define the Discovery request, specifying the manually SQL table on which the discovery should be performed. For instance

```
SELECT DISTINCT [MRP_PROPERTIES] FROM MY_TABLE
```

## 5.4 Generating a Customization Kit using the SQL Toolkit command line

If a SQL SA Toolkit customization project has previously been saved, it is possible to automatically generate the SA Customization kit without using the Graphical User Interface.

The command line is available at the following location:

### On Unix:

<SQM Installation Directory>/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.sh

### On Windows:

<SQM Installation Directory>\ServiceAdaptersToolkit\Sql\v1_2\bin\
temip_sc_start_sqltk.bat

To launch the Command line in interactive mode:

```
# temip_sc_start_sqltk[.sh .bat] –clui
```

The following commands are available at the Command Line prompt:

| | |
|---|---|
| `audit` | Audit/Check mapping of the current loaded project |
| `kit` | Generate the kit of the current loaded project |
| `load <filename>` | Load a project |
| Quit | To exit |

To launch the Command line and provide a script file to execute:

```
temip_sc_start_sqltk[.sh .bat] -file buildkit.txt
```

with buildkit.txt look like:

```
# load project
load d:\Documents and Settings\smith.EMEA.000\My
Documents\PerfSA.sqltk
# generate kit
kit
# quit
quit
```

# Chapter 6

# Request per DFD

The goal of the request per DFD feature is to minimize the number of SQL requests performed by the SQL SA, to reduce the impact on the database and to improve the collection performance.

To reach this goal, the SQL SA performs one request per data definition (Data Feeder Definition), instead of one request per measurement point (Data Feeder Instance).

By default the SQL SA executes one SQL request per DFI at each polling period. The collection performance thus depends on:

- The number of DFIs
- The number of rows to collect
- The complexity of the SQL request
- The number of connections to the database

With the request per DFD, the SQL SA executes one SQL request per DFD at each polling period as well. The collection performance thus depends on:

- The number of rows to collect
- The complexity of the SQL request

**Note**

For requests per DFD, the associated SQL request is more complex.

The SQL SA still gathers the changes that occur since the last successful polling.

## 6.1 What is the request per DFD?

The SQL SA performance improvement implies performing a single request per DFD. At each polling period, the Service Adapter performs a single SQL request per DFD and then dispatches request results to the right DFIs.

The SQL request per DFD is quite different from the request per DFI. To support a request per DFD, this request must be compliant with the following rules:

1. The ‘WHERE’ clause of the request can no longer be specified. It means that it is no longer possible to determine which rows of the SQL table must be retrieved (these rows are identified by the time window, [LAST_TIMESTAMP ... CURRRENT_GMT_TIME], when performing a request per DFI)
2. The SQL request returns a result set. This result set is composed of rows which must be structured as:
    i. DFI properties part of MRP

ii. DFI parameter values

iii. A timestamp [optional] (when the timestamp is not provided, this timestamp is SA collection time)

iv. DFI customer data [optional]

3. The SQL request must only return rows referencing monitored DFIs (unlocked DFIs). Therefore the request has to determine which DFIs are monitored. This requirement will be discussed later in the document.

4. The SQL request must determine which rows must be retrieved:

    i. Rows retrieved in a previous SQL request execution (previous polling period), must not be retrieved in a current request execution (current polling period). Therefore the requests must always maintain (persistent) the ID (timestamp) of the last retrieved (processed) row.

    ii. The SQL request is able to handle failover cases. In case of interruption of the SQM collection processing, lost of connection with the database, the request is able to determine the non-collected rows during the failing period (this case is detailed later in the document).

These conditions on the SQL request imply that the targeted SGBD supports a procedural language such as PL/SQL on Oracle. Indeed, it is necessary to perform multiple actions per SQL request:

- Perform a SELECT request to retrieve the data
- Mark the retrieved rows (marks rows on the targeted SQL table or in a dedicated table)
- Handle the monitored DFIs

=> the database must support the following kind of SQL request

SELECT [PARAMETER], [MRP_PROPERTIES] [customer parameters], [TIMESTAMP]]> FROM MyFonction()

Where the function interface is:

FUNCTION <function name>(<parameters>) RETURN TABLE.

This solution has only been validated on Oracle database (9.0.x). Consequently, the support of the SQL request per DFD is only supported on Oracle.

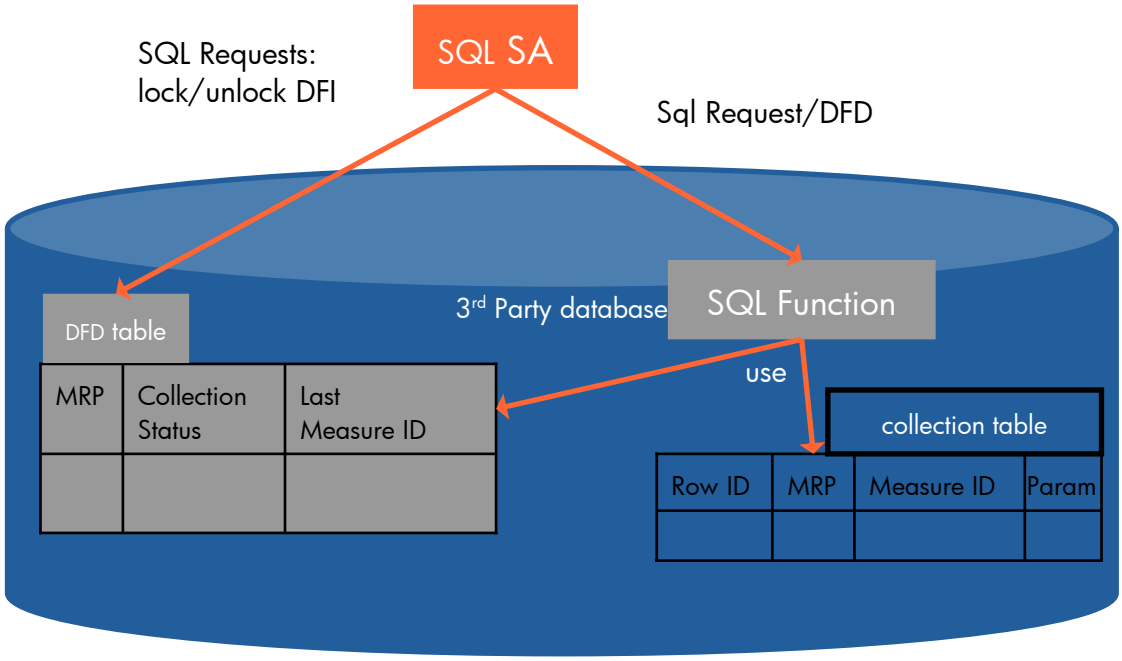
As the function needs to determine the monitored DFIs, it is necessary to define on the third party database a table to store the DFIs states. This table is managed by the SQL Service Adapter which is enhanced to perform dedicated SQL requests when a DFI is unlocked and locked. Two new hooks for defining these SQL requests are provided in the Service Adapter property file:

- An Unlock request, for instance:
    - INSERT INTO <collection state table> VALUES([MRP_PROPERTIES])
- Lock request, for instance:
    - DELETE FROM <collection state table> WHERE [MRP_PROPERTIES]

**Note**

The lock/unlock request could result in a performance degradation at SA startup and when a SQM Service is suspended. As in this situation the (lock/unlock) requests are indeed performed per DFIs.

The collection status table can also be used for storing per DFI the last measure identifier.




To sum up a SQL Service Adaptor customization performing a single SQL request per DFD is:

- a standard SQL SA customization with the SQL request per DFD enabled calling a SQL function
- a SQL script file, creating the SQL table for monitoring the DFI states
- a SQL script file, defining the SQL function used to perform data collection. This SQL function used for performing SQL request/DFD is responsible for:
    - Collecting data on the monitored DFIs
    - Marking the retrieved rows already retrieved

## 6.2 Specific customization

The DFD Collection request, the DFI Lock and Unlocked requests are specified in the Service Adapter property file, located in:

$TEMIP_SC_HOME\ServiceAdapters\Sql\v1_2\<SACustomName>_<SAversion>\properties\TeSC<SACustomName>.properties

A flag in this property file indicates if the Service Adapter must perform a request per DFI or per DFD. It is not possible to mix both modes. When the flag is not set, SQL request are performed per DFI.

***Request mode (DFD | DFI)***

Sql.DFDQuery.Mode=true

**CAUTION**

Even when running in request per DFD mode, the property file has to contain the **Sql.DefaultQuery.Pattern** property, as generated by the SQL SA Toolkit. Therefore, when you perform your SQL SA customization, using the SQL SA Toolkit, provide as table, the KPI table name! The record table KPI_TABLE, used for the request per DFD should only be provided by hand within the property file, once the SQL SA has been generated. (Confer later details)

In case the **Sql.DFDQuery.Mode** variable not defined, or with a wrong value the request per DFI mode is activated.

***Default DFD Request (global request used by all the defined DFDs)***

Sql.DFDDefaultQuery.Pattern=SELECT [MRP_PROPERTIES], [PARAMETERS], [SUBSCRIBER], [SUBSCRIBER_DOMAIN], [TIMESTAMP] FROM <Fonction name>()

***DFD Request per DFD***

Sql.DFDQuery.Pattern.<DFD Name>.<DFD Version>=SELECT [MRP_PROPERTIES], [PARAMETERS], [SUBSCRIBER], [SUBSCRIBER_DOMAIN], [TIMESTAMP] FROM <Fonction name>()

***Default DFD State Requests (lock/unlock requests)***

These requests must be specified per DFD

Sql.DFIUnlockQuery.Pattern.<DFD Name>.<DFD Version>=INSERT INTO <collection state table> VALUES ([MRP_PROPERTIES])

Sql.DFILockQuery.Pattern.<DFD Name>.<DFD Version>=DELETE FROM <collection state table> WHERE [MRP_PROPERTIES]

To help the integration of the DFD request, at set of SQL scripts will be provided, as example, here after.

## 6.3 What does it change?

The Service Adapter keeps the same processing. The Sql Request per DFD is performed in background mode, independently of the DFI collection scheduling. Request execution is scheduled at regular polling period. The request polling period is the same as the DFI collection polling period.

The SQL request execution is scheduled once a DFI is unlocked (only for the first DFI unlocked). But a delay is set before starting the request execution at regulator period. Indeed it is necessary to wait that all or at least some monitored DFIs have been unlocked before executing the SQL request. The delay is configurable per DFD in the property file with the property:

Sql.DFDQuery.Delay.<DFD Name>.<DFD Version>=<delay in ms>

When unlocking a DFI, a dedicated SQL request is executed on the database to update the table managing the DFI collection state. The function called by the SQL request uses this dedicated table for retrieving DFIs rows. And so it is necessary to wait a given elapsed time for updating the DFI collection state table before executing the collection request.

This processing does not guarantee a result for each DFI at the first polling period, but at the second polling period we are sure to get the expected results (without loosing any data).

The SQL request per DFD is stopped once all DFI are locked.

**Note**

The SQL Service Adapter therefore uses two delays:

- Sql.DFDQuery.Delay.<DFD Name>.<DFD Version>, provided within the properties file
- pollingPeriod (§4.6.2)

## 6.4 What are the SQL queries supported?

The SQL Service Adapter supports the same kind of request for the request per DFD:

1. SELECT <column name>[,<column name>]* FROM <table name> | <sql_function> WHERE <conditions>
   *Note:* conditions no more on MRP properties and select clause includes MRP properties in addition to the DFD parameters
   *Example from Model12 SA:*
   SELECT [MRP_PROPERTIES], [PARAMETERS], [SUBSCRIBER], [SUBSCRIBER_DOMAIN], [TIMESTAMP] FROM TABLE(CAST (SQM_SA_PKG.DFD_1_F() AS DFD_1_DFI_KPI_TABLE))

The SQL Service Adapter supports the following kind of requests for the lock | unlock of a DFI:

1. INSERT INTO <table name> (<column name1>[, <column nameN>]*) VALUES (<value1>[,<valueN>]*)
   *Note:* same order between column names and values (MRP properties mainly)
   *Example from Model12 SA:*
   INSERT INTO DFI_COLLECTION_STATE ([MRP_PROPERTIES]) VALUES ([MRP_PROPERTIES])

2. DELETE FROM <table name> WHERE <conditions>
   *Note:* conditions on MRP properties
   *Example from Model12 SA:*
   DELETE FROM DFD_1_DFI_STATE WHERE [MRP_PROPERTIES]

3. MERGE INTO <table_name> USING dual ON (<column name>[and <column name>]*) WHEN NOT MATCHED THEN <INSERT syntax> WHEN MATCHED THEN <UPDATE syntax>
   *Example from PerfDB SA:*
   MERGE INTO PERFDB_COLLECTION_STATE USING dual ON ( [MRP_PROPERTIES]) WHEN NOT MATCHED THEN INSERT ([MRP_PROPERTIES],COLLECTION_STATE) VALUES ([MRP_PROPERTIES],1) WHEN MATCHED THEN UPDATE SET COLLECTION_STATE=1

4. UPDATE <table name> SET <column name>=<value>[,<column name>=<value>]* WHERE <conditions>
   *Note:* conditions on MRP properties
   *Example from TeMIP Fault Statistics SA:*
   UPDATE MANAGEDOBJECTS SET DFISTATE=1 WHERE [MRP_PROPERTIES]

Of course the combination of these requests for the lock and unlock requests depends on the integrator, especially what are the tables supported and the permissions allowed in the database (no collection state table, one for both lock and unlock, one for each, existing vs new one, restricted access…).

When no table is defined to handle the DFI lock and unlock mechanism in the database or if there is no request defined for the lock and unlock of a DFI of a given DFD, the SQL SA logs then some messages to indicate that the processor of the requests failed to initialize but the SQL SA still collects.

It is neither supported nor recommended for several reasons such as performance degradation, bandwidth usage or pertinence of the data collected if they are not processed by the SQL SA…

## 6.5 As a typical example

This chapter provides an example of implementation of SQL request per DFD.

Take an example for the SQL table containing the KPIs to retrieve is structured as:

KPI TABLE

| ROWID | HOSTNAME | CPU_USE | TS |
|---|---|---|---|
| | | | |

- [ROWID: an Oracle index identifying a row]
- HOSTNAME: property part of MRP defined in the DFD (it is recommended to create indexes on these properties)
- CPU_USE: the KPI to retrieve
- TS: measure timestamp.

An additional SQL table is required to maintain the DFI collection states. This table is updated on one hand by the Service Adapter, upon DFI unlock /lock operations, and on the other hand read by the SQL request per DFD, through the associated PL/SQL function call:

COLLECTION_STATE TABLE

| ROWID | HOSTNAME | LAST_TIMESTAMP | DFI_STATE |
|---|---|---|---|
| | | | |

- [ROWID: an Oracle index identifying a row]
- HOSTNAME: property part of MRP defined in the DFD (it is recommended to create indexes on these properties)
- LAST_TIMESTAMP: the timestamp of the last KPI (row Id) returned by the PL/SQL function.
- DFI_STATE: flag, indicating whether the DFI identified by the hostname (MRP) is currently locked or unlocked.
    - 1 -> UNLOCKED
    - 0 -> LOCKED

The SQL Service Adapter is customized with the following SQL request:

- DFD request. The role of this SQL request is to:
    - Data Collection
    - Mark the collected data

Sql.DFDQuery.Pattern.<DFD Name>.<DFD Version>=SELECT [PARAMETER], [MRP_PROPERTIES],[TIMESTAMP] FROM TABLE(CAST (SQM_SA_PKG.COLLECT() AS KPI_TABLE))

**CAUTION**

Even when running in request per DFD mode, the property file has to contain the **Sql.DefaultQuery.Pattern** property, as generated by the SQL SA Toolkit. Therefore, when you perform your SQL SA customization, using the SQL SA Toolkit, provide as table, the KPI table name! The record table KPI_TABLE, used for the request per DFD should only be provided by hand within the property file, once the SQL SA has been generated. (Confer later details)

In case the **Sql.DFDQuery.Mode** variable not defined, or with a wrong value the request per DFI mode is activated.

- Unlock request, used to indicate to the DFD request which are the monitored DFIs:

```
Sql.DFIUnlockQuery.Pattern.<DFD Name>.<DFD
Version>=INSERT  INTO COLLECTION_STATE VALUE
([MRP_PROPERTIES],1,NULL)
```

- Lock request, used to indicate to the DFD request that a DFI is no longer monitored

```
Sql.DFILockQuery.Pattern.<DFD Name>.<DFD
Version>=DELETE FROM COLLECTION_STATE WHERE
[MRP_PROPERTIES]
```

Here is the SQL that could be used to create these two tables:

```
CREATE TABLE KPI (HOSTNAME VARCHAR2(36), CPU_USE
FLOAT, TS TIMESTAMP);
CREATE TABLE COLLECTION_STATE ( HOSTNAME VARCHAR2(36),
LAST_TIMESTAMP TIMESTAMP, DFI_STATE NUMBER(1));
```

The PL/SQL package implementing the function is:

Package declaration (must be put in a sql script in ‘database’ directory of the SQL SA customization working area):

```
CREATE OR REPLACE TYPE KPI_RECORD AS OBJECT (
  HOSTNAME  VARCHAR2(36),
  CPU_USE   FLOAT,
  TS TIMESTAMP);

CREATE OR REPLACE TYPE KPI_TABLE IS TABLE OF
KPI_RECORD;

CREATE OR REPLACE package SQM_SA_PKG as
    TYPE REF_CURSOR IS REF CURSOR;
    FUNCTION COLLECT() RETURN KPI_TABLE;
END SQM_SA_PKG;
```

Package body (must be put in a sql script in ‘database’ directory of the SQL SA customization working area):

```
CREATE OR REPLACE package body  SQM_SA_PKG as
FUNCTION  COLLECT () RETURN  KPI_TABLE IS  pragma
autonomous_transaction;

  single_row KPI_RECORD;
  ccur REF_CURSOR;
```

```
  retreived_rows KPI_TABLE;
  maxtime KPI.TS%TYPE;
  curhost KPI.HOSTNAME%TYPE;
BEGIN
  -- Perform the data collection using an OUTER JOIN
between
  -- KPI and COLLECTION_STATE tables
  OPEN ccur FOR
    SELECT CAST (MULTISET(
      SELECT KPI.HOSTNAME,KPI.CPU_USE,KPI.TS
      FROM KPI,COLLECTION_STATE
      WHERE KPI.HOSTNAME=COLLECTION_STATE.HOSTNAME
(+)
          AND COLLECTION_STATE.DFI_STATE=1
        AND (KPI.TS > COLLECTION_STATE.LAST_TIMESTAMP
OR COLLECTION_STATE.LAST_TIMESTAMP IS NULL)
    ) AS KPI_TABLE ) FROM DUAL;
  FETCH ccur INTO retreived_rows;
  CLOSE ccur;

  -- For each DFI Store the greater retreived
timestamp in
  -- in the COLLECTION_STATE TABLE
    OPEN ccur FOR
    SELECT HOSTNAME,MAX(TS)
    FROM  TABLE(CAST(retreived_rows AS KPI_TABLE))
    GROUP BY HOSTNAME;
  LOOP
    FETCH ccur into curhost,maxtime;
    EXIT WHEN ccur%NOTFOUND;

      UPDATE COLLECTION_STATE
       SET COLLECTION_STATE.LAST_TIMESTAMP =maxtime
       WHERE COLLECTION_STATE.HOSTNAME=curhost;

  END LOOP;
  CLOSE ccur;
  COMMIT;
  RETURN retreived_rows;
END;
END  SQM_SA_PKG;
```

## 6.6 How to build with the SQL SA Toolkit v1.2?

The SQL SA Toolkit is not designed for customizing a SQL Service Adapter implementing a SQL request per DFD.

In this context the customization process will require manual customization steps.

The whole process will be:

1. Launch the SQL SA Toolkit
2. Standard customization of the Sql Service Adapter
3. Once the DFD is fully designed

    a. replace the SQL table name defined at DFD level by the SQL procedure name (PL/SQL function name)
    b. edit the SQL request to remove the WHERE clause and to add the [MRP_PROPERTIES] keyword in the SELECT clause

4. Put in the directory ‘database’ of SA project area, the necessary SQL scripts to be pushed on the SQL database. It is necessary to write SQL scripts for:
    a. The PL/SQL function call by the Service Adapter
    b. The creation of the SQL table dedicated to DFI collection status
5. Generate the SQL SA kit
6. Unzip the kit in a temporary directory
7. Update manually the SQL SA property file to set the properties enabling the SQL request per DFD mode and defining the SQL requests performed on DFI unlock and lock events
8. Re-zip the kit
9. Install the SQL SA kit

# Appendix A

# Installation Directory Structure

The following directories and files are installed by the SQL SA Toolkit:

```
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.sh.templ
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.sh
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.ico
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.bat.templ
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_start_sqltk.bat
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_setup_sqltk.sh
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_setup_sqltk.pl
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_setup_sqltk.ico
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/bin/temip_sc_setup_sqltk.bat
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/SaSqlToolkitMtLogging.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/SaSqlToolkitMtLogging.properties.templ
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/TeSC_SqlTk_Messages.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/TeSC_SqlTkClui_Messages.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/SaSqlToolkitTraceLogging.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/config/SaSqlToolkitTraceLogging.properties.te
mpl
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/jar/TeSCSQLTK.jar
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/jar/TeSCSQLTKDataModel.jar
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/jar/TeSCSQLTKGui.jar

TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/properties/SaSqlTkKitting.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/ConnectedModeIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/CriticalMsgLevelIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/DisconnectedModeIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/InfoMsgLevelIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/SaSqlToolkitGuiResources.properties
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/SaSqlToolkitMtLogMessages.propertie
s
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/SqlSaToolkitIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/resources/WarningMsgLevelIcon.gif
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/.com.zerog.registry.xml
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/Uninstall/SqlTk.exe
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/Uninstall/Uninstall
SqlTk.lax
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/Uninstall/uninstaller.ja
r
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/resource/iawin32.dll
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/resource/remove.exe
TEMIP_SC_HOME/ServiceAdaptersToolkit/Sql/v1_2/UninstallerDataSqlTk/resource/ZGWin32LaunchHe
lper.exe
```

The following directories and files are installed by the SQL SA Customization:

TEMIP_SC_HOME/adapter/bin/***<SACustomName>_<SAVersion>***_launch.bat
TEMIP_SC_HOME/adapter/bin/***<SACustomName>_<SAVersion>***_launch.sh
TEMIP_SC_HOME/etc/addOn/***<SACustomName>_<SAVersion>***_unix.tmpl_cfg
TEMIP_SC_HOME/etc/addOn/***<SACustomName>_<SAVersion>***_windows.tmpl_cfg
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_complete_install.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_configure.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_configure.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_create_datatree.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_create_datatree.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_create_db_view.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_create_db_view.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_discovery.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_discovery.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_discovery_tmpl.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_discovery_tmpl.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_setup_connector.bat
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/bin/temip_sc_setup_connector.sh
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/config/SaSqlDiscoveryMtLogging.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/config/SaSqlDiscoveryTraceLogging.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/config/SCPlatform_SCDirector_SCApplication.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/database/…..
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/doc/….
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/lib/…
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/properties/saname.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/properties/TeSC<SACustomName>.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/properties/TeSC<SACustomName>_Messages.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/properties/TeSCSql_Version.properties
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/properties/TeSCSqlDiscovery.xml
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/DelDFDReq_<DFDName>.<DFDVersion>.xml
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/NewDFDReq_<DFDName>.<DFDVersion>.xml
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/<SACustomName>_Connectors_data.exp
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/<SACustomName>_dfds_data.exp
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/<SACustomName>_v1_4_setup.cfg
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/repository/<SACustomName>_v1_4_template.exp
TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/***<SACustomName>_<SAVersion>***/util/….

# Appendix B

# SQL Database configuration requirements

To take advantage of the SQL SA Toolkit connected mode feature and allow the Toolkit to connect to a database and extract the tables and columns definitions, the following steps have to be performed:

### For Sybase

Log on the sybase server, and install one of the following stored procedures:

sql_server.sql installs stored procedures on Adaptive Server Enterprise versions 11.9.2 and earlier.

sql_server12.sql installs stored procedures on Adaptive Server Enterprise version 12.0.

sql_server12.5.sql installs stored procedures on Adaptive Server Enterprise versions 12.5.x.

sql_asa.sql installs stored procedures on SQL Anywhere and Adaptive Server Anywhere databases.

Example:

Here is the command to install the required stored procedures within the Sybase database:

```
${SYBASE_HOME}/bin/isql -Usa -Smy_sybase_server_name -
Dmy_database_name -i${JDBC_HOME}\sp\sql_server.sql
```

Sybase documentation:

http://manuals.sybase.com/onlinebooks/group-jc/jcp0550e/jconnig/@Generic__BookTextView/928;pt=928#X

### For Oracle

Log on the oracle server as oracle user (being member of dba).

```
set ORACLE_SID=my_oracle_database_instance_name

sqlplus /nolog

connect / as sysdba

@${ORACLE_HOME}/rdbms/admin/catalog.sql

@${ORACLE_HOME}/rdbms/admin/catproc.sql
```

# Appendix C

# SQL Discovery request file example

This file defines the SQL Discovery request for each SQL SA customized DFD. This file has been created by the SQL SA Toolkit from the user property mapping.

Once the SQL SA Customization is installed, it is located in:

### On Unix:

$TEMIP_SC_HOME/ServiceAdapters/Sql/v1_2/<***SACustomName***>_<***SAversion***>/properties/TeSCSqlDiscovery.xml

### On Windows:

%TEMIP_SC_HOME%\ServiceAdapters\Sql\v1_2\<***SACustomName***>_<***SAversion***>\properties\ TeSCSqlDiscovery.xml

It has the following syntax:

```
<?xml version="1.0" encoding="UTF-8"?>
<Discoveries>
    <Discovery dfd.name="PerfDFD" dfd.version="v1_2"
        mrp.name="%SystemName%"
        query="select distinct SYSTEMNAME, IPADDRESS from PERFDB"
sa.name="SystemPerfSA" sa.version="v1_2" sql.version="v1_2">
        <Properties>
            <sc:PropertyFilter datatype="String" property.column="1"
property.name="SystemName"/>
            <sc:PropertyFilter datatype="String" property.column="2"
property.name="IPAdress"/>
            <sc:PropertyValue datatype="String"
property.name="SqlConnector">PerfConnector</sc:PropertyValue>
        </Properties>
     </Discovery>
</Discoveries>
```

Each declared property is defined, indicated in which order the property appears in the discovery request output.

The **SqlConnector** property is mandatory for the SQL SA (it does not appear at the Graphical User Interface).

# Appendix D

# 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_2"
        dfi.id="PerfDF_835227133"
mrp.name="host1.vbe.cpqcorp.net"
        sa.name="PerfSA" sa.version="v1_2"
sai.id="slmv12_acquisition_myPerf"/>

    <DFIEntry dfd.name="PerfDFD" dfd.version="v1_2"
        dfi.id="PerfD__151287840"
mrp.name="host2.vbe.cpqcorp.net"
        sa.name="PerfSA" sa.version="v1_2"
sai.id="slmv12_acquisition_myPerf"/>

    <DFIEntry dfd.name="PerfDFD" dfd.version="v1_2"
        dfi.id="PerfDF_849885112"
mrp.name="host3.vbe.cpqcorp.net"
        sa.name="PerfSA" sa.version="v1_2"
sai.id="slmv12_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 E

# 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:

### On Unix:

$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAversion>/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/Sql/v1_2/<SACustomName>_<SAversion>/discovery/filter/filter.pl -in $RAWFILE -out $FILTERFILE

status=$?

echo "Filtering completed."

exit $status
```

### On Windows:

%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2***<SACustomName>_<SAversion>***\discovery\filter***<platform name>_<director name>_<application name>***_filter.bat

```
@echo off
REM Usage:
REM   %1%: raw file
REM   %2%: filtered file
set RAWFILE=%1%
set FILTERFILE=%2%
REM Execute the perl discovery filter
perl
"%TEMIP_SC_VAR_HOME%”\ServiceAdapters\Sql\v1_2\<SACustomName>_<Savers
ion>\discovery\filter\filter.pl -in %RAWFILE% -out %FILTERFILE%

set status=%ERRORLEVEL%
echo "Filtering completed."

exit %status%
```

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

### On Unix:

```
$TEMIP_SC_VAR_HOME/ServiceAdapters/Sql/v1_2/<SACustomName>_<SAv
ersion>/discovery/filter/filter.pl
```

### On Windows:

```
%TEMIP_SC_VAR_HOME%\ServiceAdapters\Sql\v1_2\<SACustomName>_<SA
version>\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 F

# Troubleshooting

The SQL SA Toolkit logging and tracing is done in the TEMIP_SC_VAR_HOME directory if this variable was defined at the SQL SA Toolkit 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:

```
SQMSaSqlToolkit_<ID>.log
```

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

# Appendix G

# 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 |
| 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 |


invent


Java™
COMPATIBLE