HP OpenView
Service Quality Manager

Sampling Scheduling Guide

Edition: 1.4

for the HP-UX and Microsoft Windows Operating Systems

March 2007

Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

Trademark Notices

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

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

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

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

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

UNIX® is a registered trademark of The Open Group.

Contents

Preface

This document presents a new OpenView SQM calculation scheduling mode called “Sampling Scheduling”.

It gives an overall view of this new feature.

Intended Audience

This document addresses the following audience:

- Solution architects and service designers
- Service operators

Prerequisite Reading

This document assumes that you have read the *OpenView Service Quality Manager Overview*.

Supported Software

The term UNIX is used as a generic reference to the operating system, unless otherwise specified.

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

Product Version	**Operating System**
OpenView Service Quality Manager 1.4	HP-UX 11.11 Windows XP

Typographical Conventions

`Courier` Font:

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

Italic Text:

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

Bold Text:

- To introduce new terms and to emphasize important words.

Associated Documents

The following OpenView SQM documentation can be useful to understand the Sampling Scheduling:

- *OpenView SQM SLA Monitoring UI User's Guide*
- *OpenView SQM Service Designer UI User's Guide*
- *OpenView SQM SLA Administration UI User's Guide*
- *OpenView SQM Overview*
- *OpenView SQM Getting Started Guide*
- *OpenView SQM Information Modeling Reference Guide*

Support

You can visit the HP OpenView support web site at:

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

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

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

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

Chapter 1

Presentation of the Sampling Scheduling

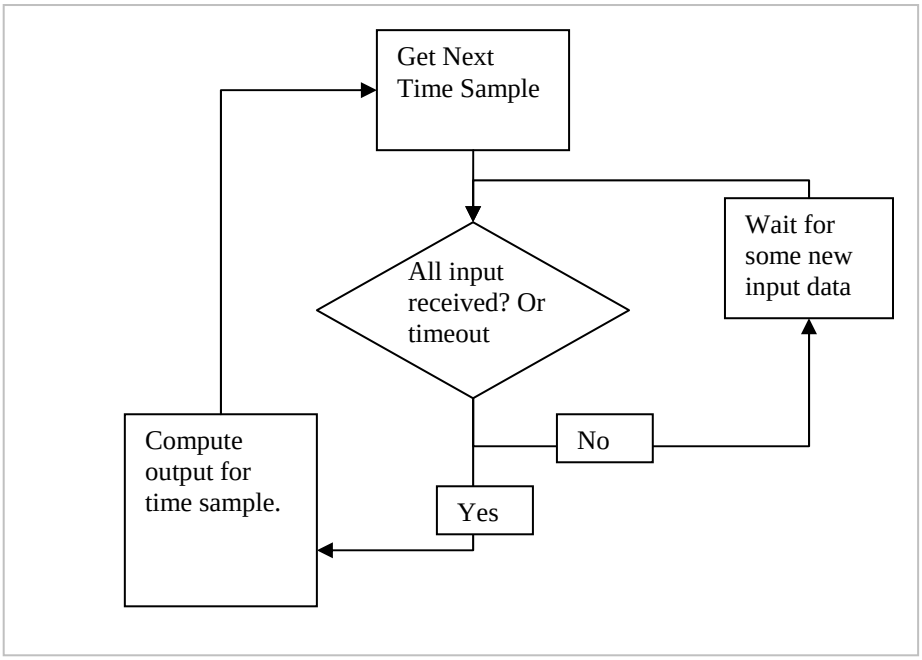
In previous versions of OpenView SQM, each service definition loaded in the calculation engine is computed with a regular period, and this period is the same for every models. This scheduling method is suitable when data are collected in real time, but it leads to some inconsistencies when data sources are not synchronized or when data are collected in batch with a delivery delay.

In order to provide a solution for these common problems, a new scheduling method called “Sampling Scheduling” has been introduced in OpenView SQM.

In sampling mode, the services are no more computed with a regular period, but the calculations are triggered for specified timestamp and according to the received input data.

The scheduling algorithm is detailed below.

Figure 1 Sampling Scheduling Algorithm



- **Time Samples**

  The time samples are the specific moment when a calculation will be executed for the service.

  Each service definition has its own time sampling properties.

In OpenView SQM, you have to select specific minutes of the hours to define the times sample.

For example if you choose “0 20 40” as time sample values, the calculation engine will trigger a calculation for the following time sample 00:00, 00:20, 00:40, 01:00, 01:20, 01:40, 02:00… and so on. Unlike in normal scheduling mode, these time samples are well specified and the algorithm will guarantee that every time sample will be computed once.

It is important to dissociate the time sample from the timestamp of the calculation because the calculation engine waits for every input of a time sample to trigger its calculation. For example, the time sample “10:20” can be computed at anytime: if the last input for this time sample has been received at 10:45, it will be computed just after, at most one minute after 10:45.

- **Sampling Parameter**

  As mentioned above, the calculation engine waits for every input of a time sample to trigger its computation.

  In fact, this is not exactly every input that is expected, but every input for some specific parameters of the model. These specific parameters are called the “Sampling Parameters”, because they have a specific role in the scheduling of the service.

  Since the calculation engine expects a value for every sampling parameter of the model for every time sample, **the collection layer (Service Adapter) must be adapted to this scheduling mode and ensure that a value is effectively published for every sampling parameter, for every DFI and for every time sample.**

- **Timeout**

  A complete time sample is a time sample that has received every expected input.

  An incomplete time sample is a time sample that has not received every input. In order to prevent from endless waiting of the input for an incomplete time sample, the computation can be triggered when some timeouts expire.

  Two different timeouts can be defined at service definition level (each service can have its own timeout values).

  Global Time Out:

  If (Current Timestamp - Time Sample > Global Time Out), then the Time Sample will be computed.

  Data Timeout:

  If (Last Received input timestamp - Time Sample > Data Time Out), then the Time Sample will be computed.

  The values of these timeouts will depend on the capabilities of your collection layer and your constraints in term of monitoring.

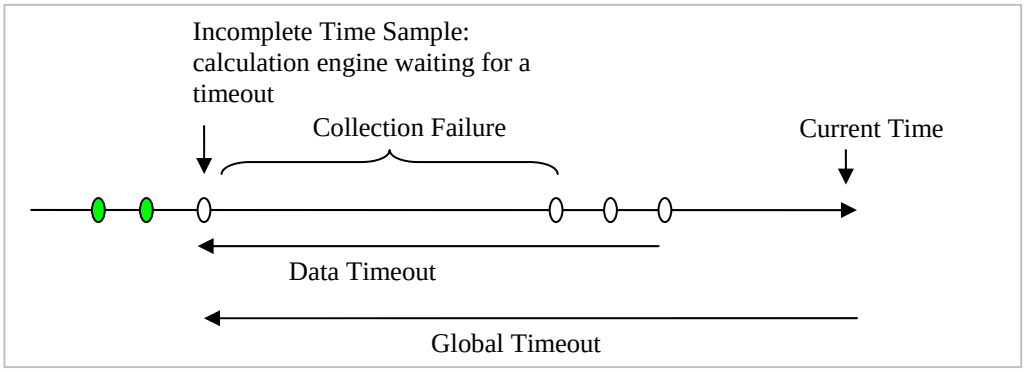
  The Global timeout defines the maximum delay of the last computed time sample. A high value (several days or endless timeout) will ensure that no computation will be done in case of a failure in the collection, but the calculation engine can accumulate a very long delay.

  The Data timeout permits to catch up the delay due to a failure of the data collection as soon as the collection is restored. It also permits to avoid a too long global timeout when the collection is incomplete (unbound SCI, locked DFI). A typical value is twice the sampling period. This means that when a collection step is started, the previous time sample will be computed if it was incomplete.

Note

In case on of the Service Adapter publish in batch (once a day for example), the Data Time out must be greater than the batch period.

Figure 2 The Timeouts



- **Time Sample Computation**

  The Time Samples are computed in chronological order. If time sample “10:00” is the next time sample to compute but it is not completed the calculation will wait for missing input for this time sample. If time sample “10:10” is completed, it will not be computed: it has to wait for the calculation of “10:00”.

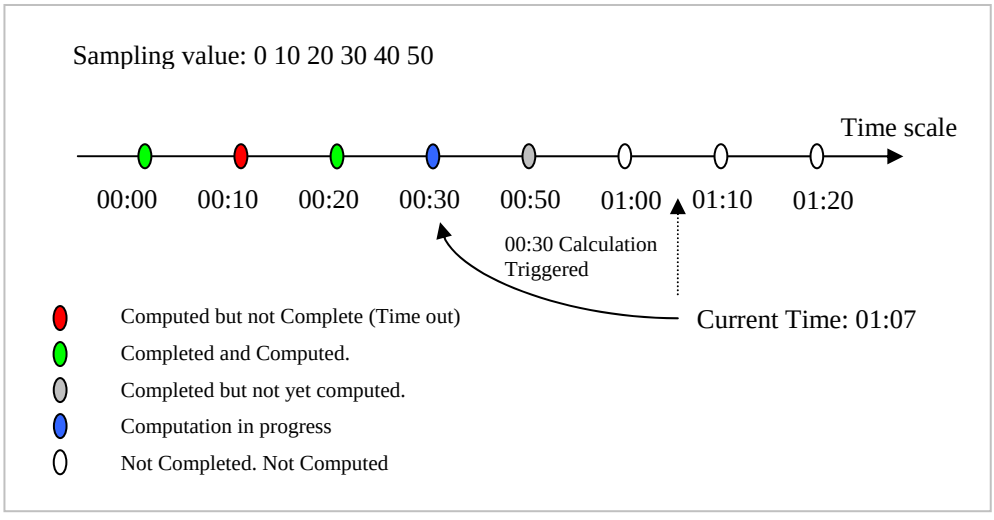
  When “10:00” is in time out or becomes complete, the calculation engine will compute it, and then will immediately compute “10:10” time sample which is already completed. If the next time sample is also completed, it will also be computed and so on.

  When a time sample is computed, the calculation engine uses only the input received for this time sample (and the older one) and **the output are published with the timestamp of the time sample.**

  The figure below shows:

  - The Time scale with the time samples defined by Sampling Values.
  - The possible state of the time samples

Figure 3 Time Sample States



- **Dataload**

  A benefit of sampling scheduling is that it is possible to load a new model (Service Definition + Instance) and perform a dataload (Load data in the past). If you set the Global Timeout to -1 (Endless Time out), the calculation will never go beyond the provided input timestamp. You can thus submit input values in the past, every past time sample will be computed when inputs are received. The computed output will correspond to the input received for every time sample.

  More details on the dataload procedure are given in section 2.4 Dataload.

Chapter 2

Sampling Scheduling Usage

This chapter describes how to design and deploy service scheduled in sampling mode in OpenView SQM.

This chapter contains the following sections:

- Section 2.1 Before You Begin
- Section 2.2 Modeling Your Service
- Section 2.3 Sampling Feature restriction
- Section 2.4 Dataload

2.1 Before You Begin

This section provides the information you need before beginning the service design phase of your sampling solution.

Before implementing a monitoring solution based on sampling scheduling, you must ensure that the collection layer (Service Adapters) can support this scheduling mode. This is described in section 2.2.3 Data Collection for sampling scheduling.

Also some restrictions apply to this feature (See 2.3 Sampling Feature restriction). You have to be aware of them and take them in account when you design your service.

2.2 Modeling Your Service

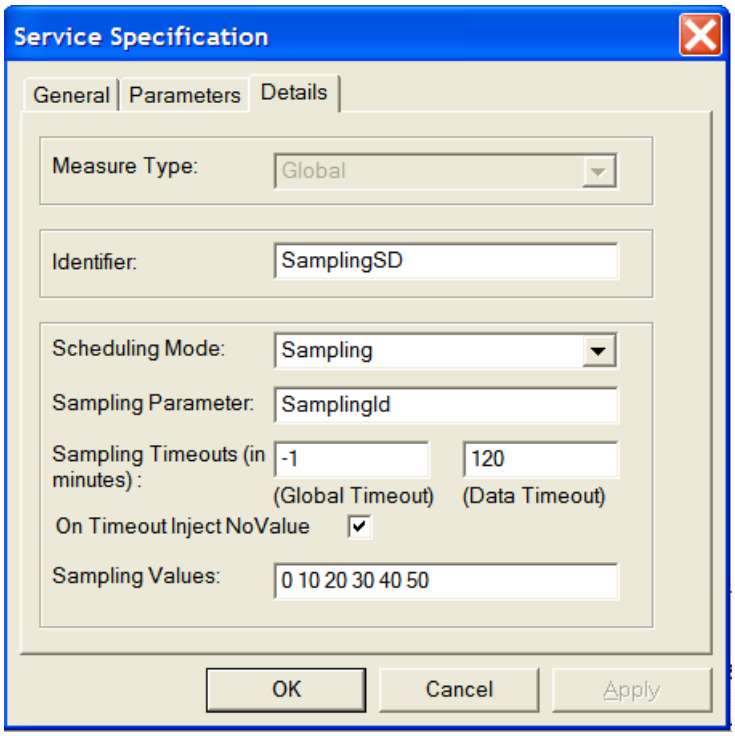
This section describes the few steps that need to be done in the service designer to define a service in sampling mode

2.2.1 Service definition properties

In Service Designer, you have to open the specification of the Service Definition. In the “Details” tab, you have to select the “Sampling” mode and then define:

- The Sampling parameter
- The Global Timeout (Left Text Box) and the Data timeout (Right Text Box). The timeout out are in minutes and -1 means endless timeout
- The “On Timeout Inject NoValue” checkbox
- The sampling values (list of number between 0 and 59)

Figure 4 Sampling Properties



Regarding the Sampling Parameter, every parameter in the service definition having its name (Identifier) starting with this value will be considered as a sampling parameter. For this reason, the provided value should be meaningful (“SampleXX”).

At least one parameter in the model must match this criterion (start with Sampling Parameter value), otherwise the service will not be scheduled in sampling mode.

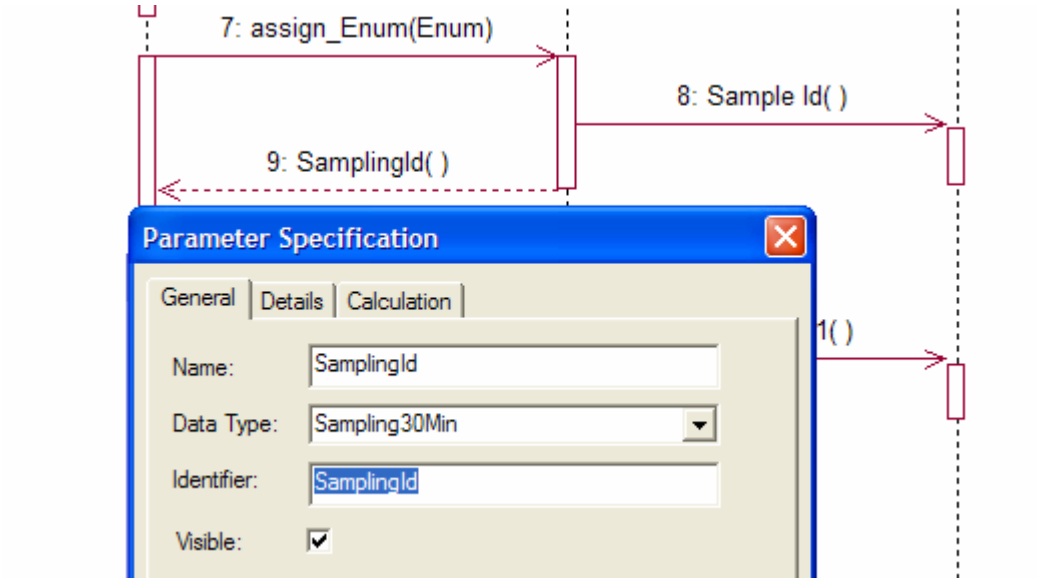
The “On Timeout Inject NoValue” checkbox is the flag to choose between inject NoValue or propagate the pervious measure, when it is true, on sample timeout, SPDM “inject” NoValue for not received measures.

The Sampling Values are used to specify what will be the time samples for your model. The Sampling Values corresponds to the minutes of the hours. For example if you choose “0 10 20 30 40 50” as time sample values, the calculation engine will trigger a calculation for the following time sample 00:00, 00:10, 00:20, 00:30, 00:40, 00:50,01:00, 01:10, 01:20… and so on.

2.2.2 Setup primary sampling parameter

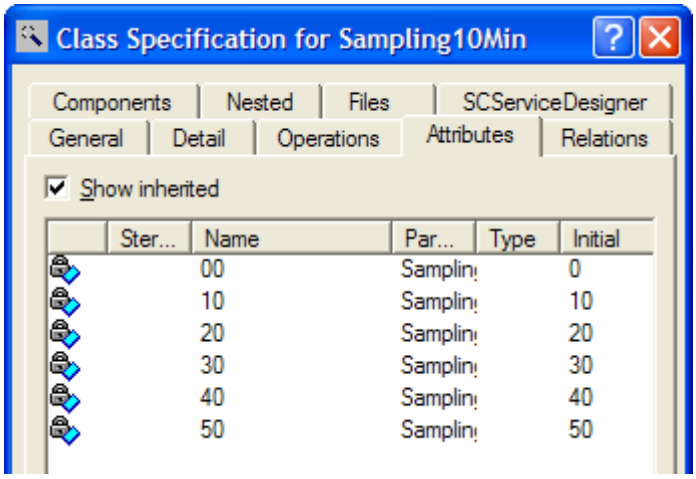
For every data source (DFD) that is supposed to collect in sampling mode, you must add the related parameter in the primary SCD of your model.
The figure below illustrates how to perform this binding (in the example the DFD sampling parameter is name “Sample Id”).

Figure 5 Sampling Parameter Binding



This sampling parameter should be of type Enum and the Enum values should be the same as the ones defined for the service definition (Sampling Values). This is not mandatory, the parameter can also be of type Int, but the usage of an Enum type provides more consistency in the model.

Figure 6 A Sample (Sampling) Enum



2.2.3 Data Collection for sampling scheduling

The collection layer must be able to publish the sampling parameter values for every DFI at every time sample defined in the service definition.

Ex: at 10:00 value 0, at 10:10 values 10, at 10:20 value 20…

Note

Data can be published late, in burst and non chronologically: at 12:00, it is possible to publish (10:00,0)(10:20,20)(10:40,40)(10:10,10)….(12:00,00).

If you are not using a SQL SA service adapter, you have to study how this can be achieved.

The next section explains how to adapt an existing SQL SA to this mode.

2.2.3.1 Normalization of data for SQLSA

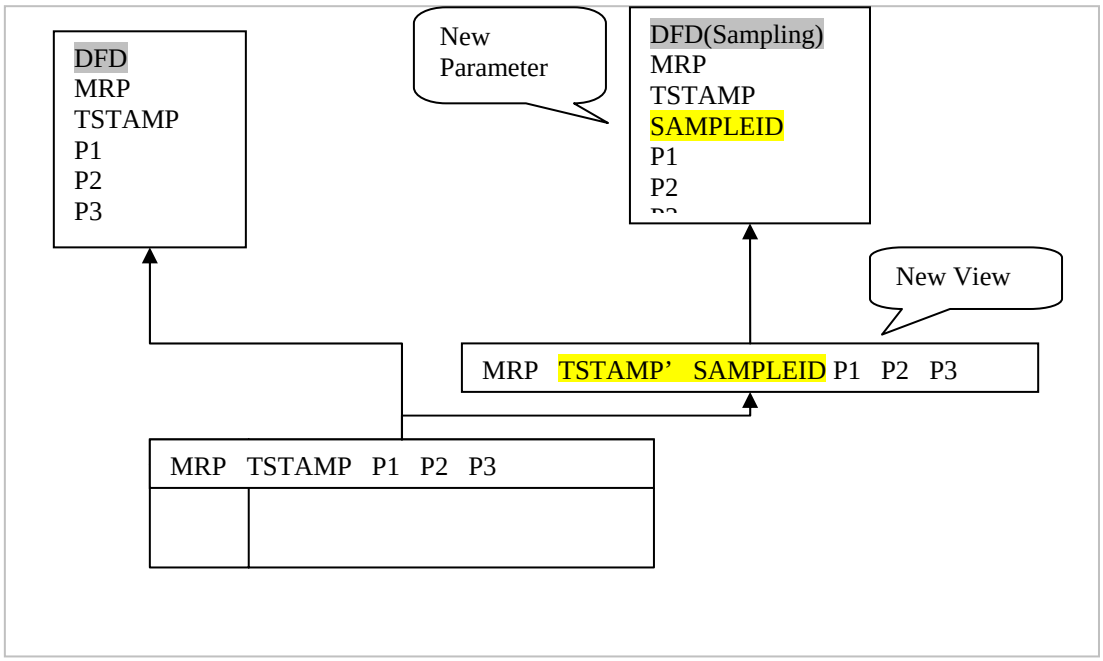
Let's consider the following situation: you collect data in a DATA table using a SQL SA customization. This table contains a MPR column, a TSTAMP column and the collected parameter (P1, P2).

The problem is that the data are delivered with a delay (every hour for example). This is a typical case where you will take benefit of the new sampling scheduling mode. For this you will have to update the collection layer, and the model.

In the model, you have to add a Sampling parameter bound to a new parameter of the DFD. This new parameter of the DFD must collect some correct sample values.

A simple way to achieve this is to collect on a new view that will normalize your input.

Figure 7 Collection Modification



Here is a sample scripts for the view creation. Adapt it to your needs.

```
CREATE OR REPLACE VIEW SAMPLING_DATA AS
  SELECT
   mrp,
   hour + SampleId/24/60 tstamp,
   SampleId,
   SUM(p1) P1,
   AVG(p2) p2  -- aggregation depends on your needs.
  FROM (
    SELECT
      mrp,
      TRUNC(tstamp,'HH24') hour,
      CEIL((TO_CHAR(tstamp,'MI'))/PERIOD) * PERIOD  SampleId,
      p1,
      p2
    FROM
    DATA
   )
  GROUP BY mrp, hour ,sampleid;
```

The PERIOD must be replaced by its correct value that depends on the sampling values. It corresponds to the number of minutes between 2 sampling values (5 or 10 or 20...). If the sampling values are not periodic, the view will not function. This particular case is not detailed in this document. You have to introduce a table with the sampling values and use it.

Your SQL SA/DFD must have a new parameter for SamplingId column and collect on SAMPLING_DATA view instead of DATA table.

You can then feed the Data table at any time. SPDM will wait for new data and process all of them when entered. Secondary parameter will be correct at any time. The only restriction is to provide at least one row for each MRP/time sample (with our without values for P1, P2…).

If you provide more that one raw per mrp/time sample, you should consider decreasing the sampling period, or use proper aggregate function depending on collected data.

It is possible to have more complex view in order to mix sampled and not sampled parameter, but this is a very particular case that should be studied depending on your need. It is not explained in this document.

2.2.4 Time Sample Completion Detection

Once you have designed a service model, there is no change in the deployment of this model. You have to load it in OpenView SQM and instantiate it like any other model.

Anyway, since the scheduling algorithm is different, the behavior of the service in the monitoring will be different.

In order to detect if a time sample is completed (all expected inputs are received), the calculation engine checks if a correct value (with a correct timestamp) has been received for every “Sampling” parameters of the model.

In fact, not every SCI are expected to have a value for a given time sample. Only the SCI involved in an SLA are considered, and the locked SCI (at the moment of the time sample) are not considered.

If the Sampling parameter of a SCD is customer dependent, the calculation engine expects to receive a value for every customers involved in an SLA for this SCI.

The detection of time sample completion after a model update (new SCI, new Cust in SLA) can also lead to time-out, because the calculation engine does not keep the history of the model, but only its current state.

2.3 Sampling Feature restrictions

2.3.1 Sharing

Local Instance sharing is fully supported.

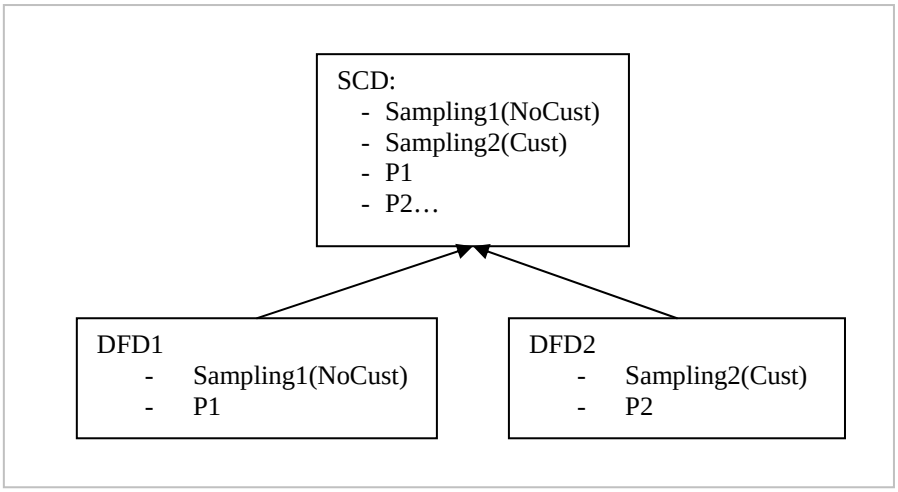
Global sharing is supported provided every service definitions sharing the same SCD are scheduled in sampling mode.

If a SCD is shared between a Service scheduled in sampling mode and in normal mode, the behavior is not predictable. Thus this use case is not recommended.

2.3.2 Datafeeder

Each SCD can manage at most one sampling parameter for customer dependent input and one sampling parameter for customer independent input.

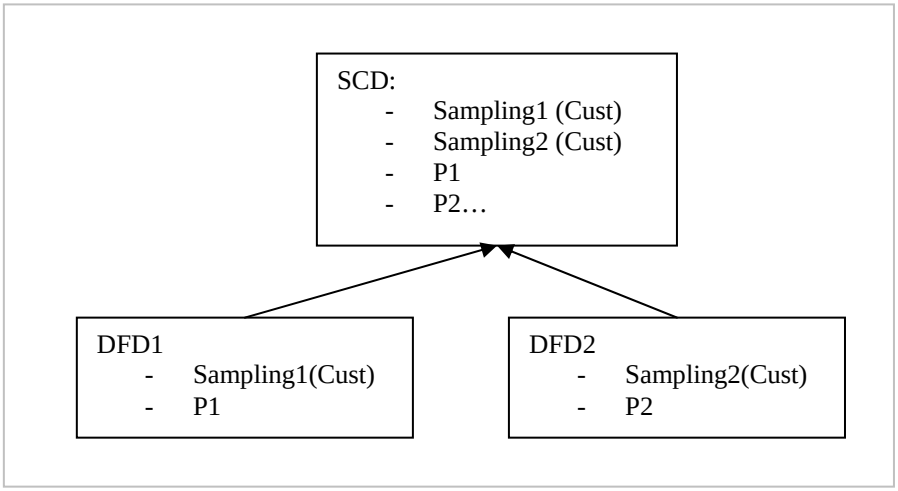
Figure 8 DFD Limitation – Supported Case



If more sampling parameters are defined, the collection engine will not expect to receive a value for each sampling parameter but only for one of them.

In the situation below, the calculation does not expect a value for Sampling1 and Sampling2, but only one of them.

Figure 9 DFD Limitation – Unsupported Case



2.3.3 Instantiation

The calculation engine does not store any information related to the DFI binding. For this reason, it expects a value for every SCI whether it bound to the DFI or not.

For this reason, every SCI involved in a sampling model has to be bound to a declared DFI. For the same reason, the DFI must not be locked manually.

2.3.4 Autoforward Parameter

Auto-forward parameters are not supported in sampling mode, so, please check there is no ‘Auto Propagate’ parameter in your sampling service definition.

2.4 Dataload

A dataload is an operation that consists in providing past-data for a model and re-computes all the history of the model using these data.

It is possible to use the property of the scheduling mode to perform a dataload properly (all output computed as expected) and safely (easy to run and control). This feature is not fully integrated in \OpenView SQM and some restrictions and manual actions are required.

First, in the current version, only dataload on new model is supported. As soon as a model has started to collect and compute data, it will not be possible to load past data on the model.

Then, unlike for Datamart, there is no tool to load data from a CSV data file. Data has to be provided in XML format on Tibco bus (either Msg 52 or 56). This can be done using a specific customization of the SQLSA or by using the CLUI command line (temip_sc_publish).

And finally, the following query has to be run in order to change sampling configuration for dataload mode:

```
UPDATE SAMPLING_CONFIGURATION SET
dataload_mode='T', sampling_timeout=-1
WHERE sd_Ref IN (SELECT id FROM SERVICE_COMPONENT_DEF WHERE
name=:sd_name);
Commit;
```

The procedure to perform a dataload properly is the following:

- Load the new model (Service Definition, all instances, all SLA)
- The service definition must be scheduled in sampling mode, the global time out must be -1 and the dataload mode must be ‘T’ (see query above)
- Publish the data history to the calculation engine. You have to ensure that the first set of data sent to the calculation engine is the older one of your history

The input data publication does not have to be in chronological order, but for performance reasons, it is better to publish them in chronological order. In case the input are not in chronological order, you have to set a high value for the data time out in order to be sure that incomplete time sample are not processed.

The Calculation engine will compute and publish output in chronological order and the Status engine will be able to process them correctly.

Once all the data are processed, you can start the regular collection with the service adapters on your model. You can even change the scheduling mode to Normal by reloading the updated service definition. In any case, it is recommended to reload the original service definition in order to restore the updated model properties (Global Timeout, Dataload Mode).

2.4.1 Parameter impacting sampling scheduling

The parameters below are stored in the spdm database (user spdm).

2.4.1.1 Sampling_configuration table

SAMPLE_COMPLETION_METHOD:

By default, the sample values must be correct (equal to the number of minutes). If you set the “SAMPLE_COMPLETION_METHOD” to “ts”, the value of the

sampling parameter will not be checked. It is just expected to receive something at correct timestamp. This flag can be useful if you want to use the sampling on an existing model to perform the dataload and then turn it back to normal scheduling.

DATALOAD_MODE:

If set to 'T' the SCI state (locked/unlocked) and creation date are ignored. All SCI are taken in account to detect complete timestamp.

SAMPLING_TIMEOUT, DATA_TIMEOUT:

It is advised to set SAMPLING_TIMEOUT to -1 and DATA_TIMEOUT to 0 if you deliver data in chronological order.

If data are not provided in chronological order, but every time samples are provided, set SAMPLING_TIMEOUT to -1 and DATA_TIMEOUT to -1.

MAX_STEP_PER_SCHEDULE:

Each minute the calculation engine will compute MAX_STEP_PER_SCHEDULE step, then publish all computed values. This value must be high enough to speed up the dataload, but not too much in order to balance computation/publish and SLOM processing. You can tune this value directly while the dataload is in progress. Start with a small value, monitor the progression (see below) and try to augment it until you reach the maximum efficiency.

ON_TIMEOUT_INJECT_NOVALUE:

If set to 'T', on sample timeout, SPDM "inject" NoValue for not received measures.

2.4.1.2 Config_parameter table

LOG_LEVEL:

If you set LOG_LEVEL to 2, you can monitor dataload progress in LOGGED_EVENT table.

Depending on the amount of data to proceed and of the size of the model, you can update the following configuration parameters to improve dataload performance:

BULK_STORE_THRESHOLD:

Set to 0. All input data will be stored using the fast storage procedure.

BULK_STORE_MAX_ROW:

You can augment this value to give more priority to the bulk storage processing and thus improve the processing of the input. On the other hand, if this value is too high, the data storage for the service scheduled in normal mode will be altered.

LATE_PUBLISH_BURST_SIZE:

Must be greater than MAX_STEP_PER_SCHEDULE.

Important note

A High value of MAX_STEP_PER_SCHEDULE and LATE_PUBLISH_BURST_SIZE can delay the processing of a model update (Service Definition) by the calculation engine. Avoid doing definition update during a dataload or reduce these two parameters before doing the update and restore high value after.

2.4.2 Disk usage / Purge

The calculation engine is not designed to store a long retention delay of data.

If you perform a dataload on a big model and for a long history, you have to monitor very closely the disk usage of the table space and you have to activate the purge once a day to spread data among different partition.

In dataload mode, the retention delay set in Tibco flag (SPDM_config/Status/DataRetentionDelay(days) and SPDM_config/Service/DataRetentionDelay(days)) is not valid and a far greater data retention is applied.

The maximum retention of data kept in the calculation engine (if purge is done once a day) is approximately (Nb Partitions - 1) * Data Processed per day.

Nb Partitions = 9 by default.

Data Processed per day (in day) = 24*60*MAX_STEP_PER_SCHEDULUE*SAMPLING PERIOD/24/60= MAX_STEP_PER_SCHEDULUE*SAMPLING PERIOD

A typical value is 10*5 = 50 Days of data processed per day.

If you purge once a day, you have to ensure that the calculation engine can hold 8*50 = 400 days of data (or the entire history to load).

If this is too much, you have to consider to slow down the dataload procedure (reduce the MAX_STEP_PER_SCHEDULUE), which is probably not the best idea, or perform a manual purge procedure more often. The negative aspect is that the retention of the other model will be affected. If you perform two purges per day, the retention is divided by 2.

The manual purge procedure is required because the calculation engine only accepts one purge per day. To force a purge manually, run:

```
UPDATE PARTITIONING SET
last_purge_tstamp=last_purge_tstamp-1;

COMMIT;
```

Then launch the purge AMI. The update and the AMI call can be done in a cron script.

Acronyms

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

Term	Description
DC	Data Collector
DFD	Data Feeder Definition
DFI	Data Feeder Instance
GUI	Graphical User Interface
OS	Operating System
QoS	Quality of Service
SA	Service Adapter
SAI	Service Adapter Instance
SAP	Service Access Point
SC	Service Component
SD	Service Definition
SI	Service Instance
SLA	Service Level Agreement
SLM	Service Level Management
SLO	Service Level
SPD	Service Parameter Definition
SPDM	Service Performance Data Manager
SRM	Service Repository Manager
SQL	Standard Query Language

Glossary

This glossary defines terminology specifically related to sampling scheduling.

Normal Scheduling

The calculation engine triggers the service calculation on a regular basis (5 minutes per default). The timestamp of the calculation is used for publication.

Sampling Scheduling

The new scheduling mode described in this document based on received input and a sampling of the time scale.

Time Sample

The specified moment of the time scale when a calculation can occur in sampling mode. In OpenView SQM they are represented by fixed minutes of the hour.

Complete/Incomplete Time sample

All / Not all the expected input has been received for the time sample.


invent


Java™
COMPATIBLE