D4.1a - VISHNU Information Management Services Module Design

COLLABORATORS			
	TITLE : D4.1a - VISHNU Information Management Services Module Design		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé	April 5, 2011	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1	06/04/2011	Deliverable version	SysFera

Contents

List of Figures

Chapter 1

Document presentation

1.1 Document objectives

This document presents the detailed internal design of the Information Management Services (IMS) module. The purpose of this module is to handle all aspects of information management within the VISHNU system. The functional and non-functional requirements for this module are those described in the referenced specification documents. The current document is part of the design phase of the software and therefore its main goal is to define the main components of the system architecture and their relationships.

1.2 Document structure

- Chapter 1 contains a brief overview of the document content.
- Chapter 2 contains a description of the IMS system architecture.
- Chapter 3 contains an overview of the IMS system modelization.
- Chapter 4 describes the internal API used for remote procedure calls through SysFera-DS.
- Chapter 5 describes the internal classes and data structures

1.3 References

- [D1.1a]: VISHNU General specifications
- [D1.1b]: VISHNU Spécifications techniques des besoins
- [D1.1c]: VISHNU API Detailed specifications
- [D1.1g]: VISHNU Technical architecture
- [D1.1g-DAT]: VISHNU Dossier d'Architecture Technique

1.4 Acronyms

- **API**: Application Programming Interface
- **CLI**: Command line interface

- **DB**: Database
- **n/a**: Not Applicable
- **IMS**: Information Management Services
- **WS**: Web Services

1.5 Glossary

- **Components**: the software components represents a library or an executable program that provides a given interface to other components or to end-users.
- **DIET**: component of the SysFera-DS solution that provides the communication layer between agents that are part of the VISHNU infrastructure.
- **GoDIET**: component of the SysFera-DS solution that provides a launcher for all processes of the VISHNU infrastructure.
- **LogCentral**: component of the SysFera-DS solution that provides event notifications within a SysFera-DS platform.
- **SeD**: A Server Daemon is a SysFera-DS agent that provides services through the SysFera-DS API.
- **Serialized type**: this is a class of data (C++ Class) which instances can be serialized in a XML string before being sent over a communication channel (CORBA for example).
- **SysFera-DS**: open-source middleware developped by SysFera.
- **VISHNU Process**: a process running permanently that provides VISHNU services. This is a sub-class of SeD.

Chapter 2

IMS System Architecture

2.1 IMS software components

We present in this section a description of the IMS module in terms of software components. In addition we show the dependencies between components to highlight their reuse. These components follow a client/server model. We present the different software layers from services (provided directly to the user) to the database (used by the server). The IMS client and server packages have been split into seven different interrelated components. The diagrams shown in sections 2.1.1 and 2.1.2 describe the relationships between these components. The definitions of the components are the following:

- **External API** contains precisely the services provided to the user as defined in the detailed specifications. We're on the client side.
- **Internal API** is the interface of the server side, used by the IMS Client components. It provides a set of services that are remotely accessible through the SysFera-DS middleware. This layer does not contain any business logic and uses directly the IMS SeD for service implementation.
- **IMS Client** contains intermediate (proxy) classes providing remote access to the business objects of **IMS SeD**.
- **IMS Server** provides the business logic of the IMS module on the server side. The set of classes in this component cover all the IMS objects that have a proxy class that publishes services to the external API, and additionnally some classes that provide internal mechanisms of the IMS module.
- **Sysfera-DS Client API** is the C++ client API provided by the SysFera-DS toolbox.
- **Sysfera-DS Server API** is the C++ server API provided by the SysFera-DS toolbox.
- **VISHNU Database** stores all data handled by the IMS SeD.

2.1.1 IMS client-side components

This diagram shows the components that compose the client side of the VISHNU IMS system and their interfaces. All the interfaces of the IMS Client component are shown (CLI, WS, python, C++).

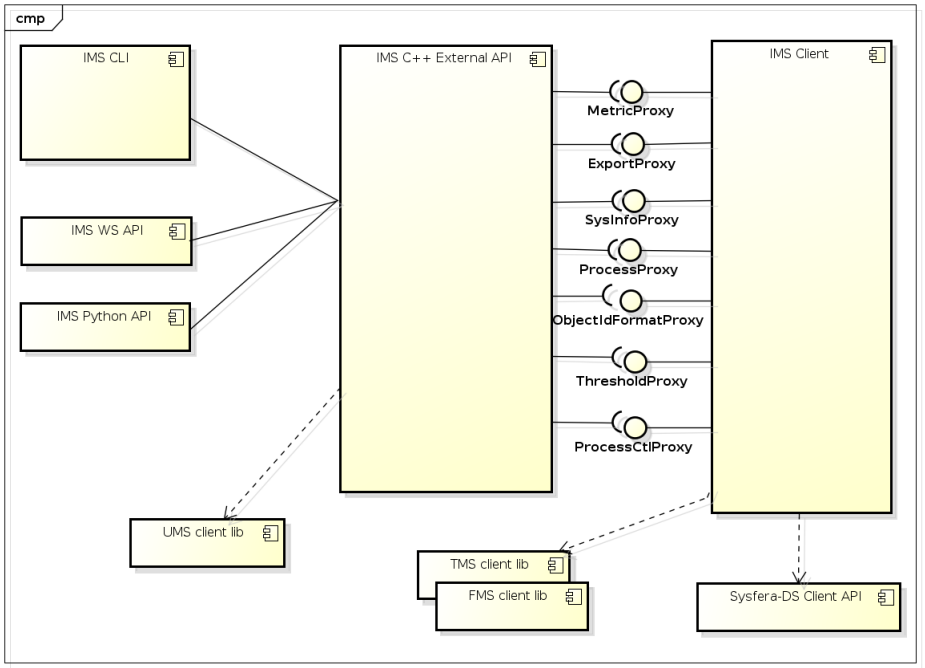

Figure 2.1: IMS client-side components

2.1.2 IMS server-side components

This diagram shows the components that compose the server side of the VISHNU IMS system and their interfaces. All the interfaces of the IMS SeD component are shown.

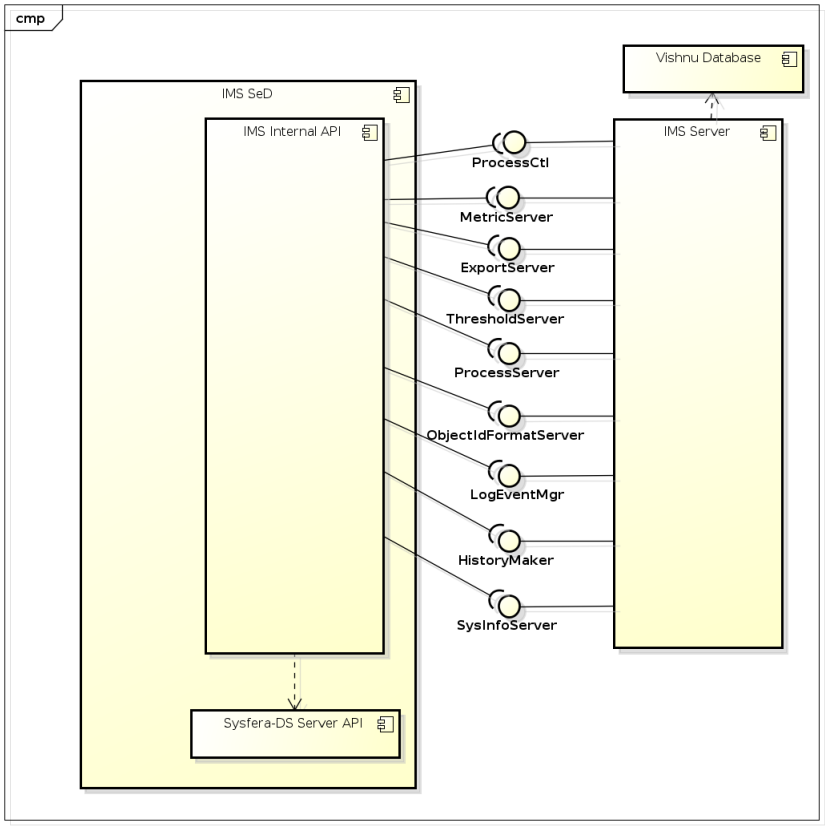

Figure 2.2: IMS server-side components

2.2 IMS package dependencies

In order to install the IMS service on the infrastructure, the following tables list all the packages that must be installed on the IMS client and on the IMS server hosts:

2.2.1 IMS Client required packages

Package Name	Description
SysFera-DS/DIET	Middleware
VISHNU UMS client	VISHNU Client Package for user and session management
VISHNU TMS client	VISHNU Client Package for tasks management
VISHNU FMS client	VISHNU Client Package for files management
Python (optional)	Python language, required to use the VISHNU Python API
Java 1.6 (optional)	Java environment, required to use the VISHNU WS API

2.2.2 IMS Server required packages

Package Name	Description
SysFera-DS/DIET	Middleware
SysFera-DS/LogService	Middleware component for monitoring
VISHNU UMS	VISHNU Package for user and session management
PostgreSQL client (or Oracle client)	Database client

2.3 Deployment aspects of IMS

We explain here how the VISHNU IMS module will be deployed on a physical infrastructure as illustrated in figures 2.3 and 2.4. Each box in the figures represents an operating system in which a component or a set of components are running as processes. For an overview of all components of the VISHNU infrastructure, please refer to document [D1.1g]. The IMS module components are the following:

- **IMS SeD** is the process that provides all IMS internal services. This process is required on any host of the VISHNU infrastructure where VISHNU processes have to be monitored. Without an IMS process running on the host it won't be possible to use IMS services on that host, i.e. restart processes, get system information, do loadshedding, etc.
- **Client host** is the host used by the user to launch IMS requests through one of the VISHNU user interfaces. It contains all the client components required to make an IMS service request.
- **LogCentral** is a SysFera-DS process that is necessary to monitor the DIET platform. This process is part of DIET in the SysFera-DS package. It can be deployed on any host that can connect through CORBA to the omniNames and DIET MasterAgent. To simplify the deployment, it can be running on the same machine as these processes.
- **SysFera-DS Bus** is the specific software layer that ensures the communication between client hosts and server hosts.
- **VISHNU database**: this component represents a unique instance of an Oracle or PostgreSQL database.

2.3.1 IMS deployment overview

The following diagram shows an example of VISHNU infrastructure with one instance of FMS Server and one instance of TMS Server. All components of the infrastructure are shown including SysFera-DS processes, the JBoss web server used to deploy the web services API, and the database server. The VISHNU Admin system is hosting the SysFera-DS/GoDIET application that is used to launch all processes of the VISHNU infrastructure using ssh connections (red arrows). The SysFera-DS LogCentral process is used to publish events when VISHNU elements are connected to or disconnected from the VISHNU infrastructure. These events are sent to VISHNU IMS Servers that are subscribers to the LogCentral event flow.

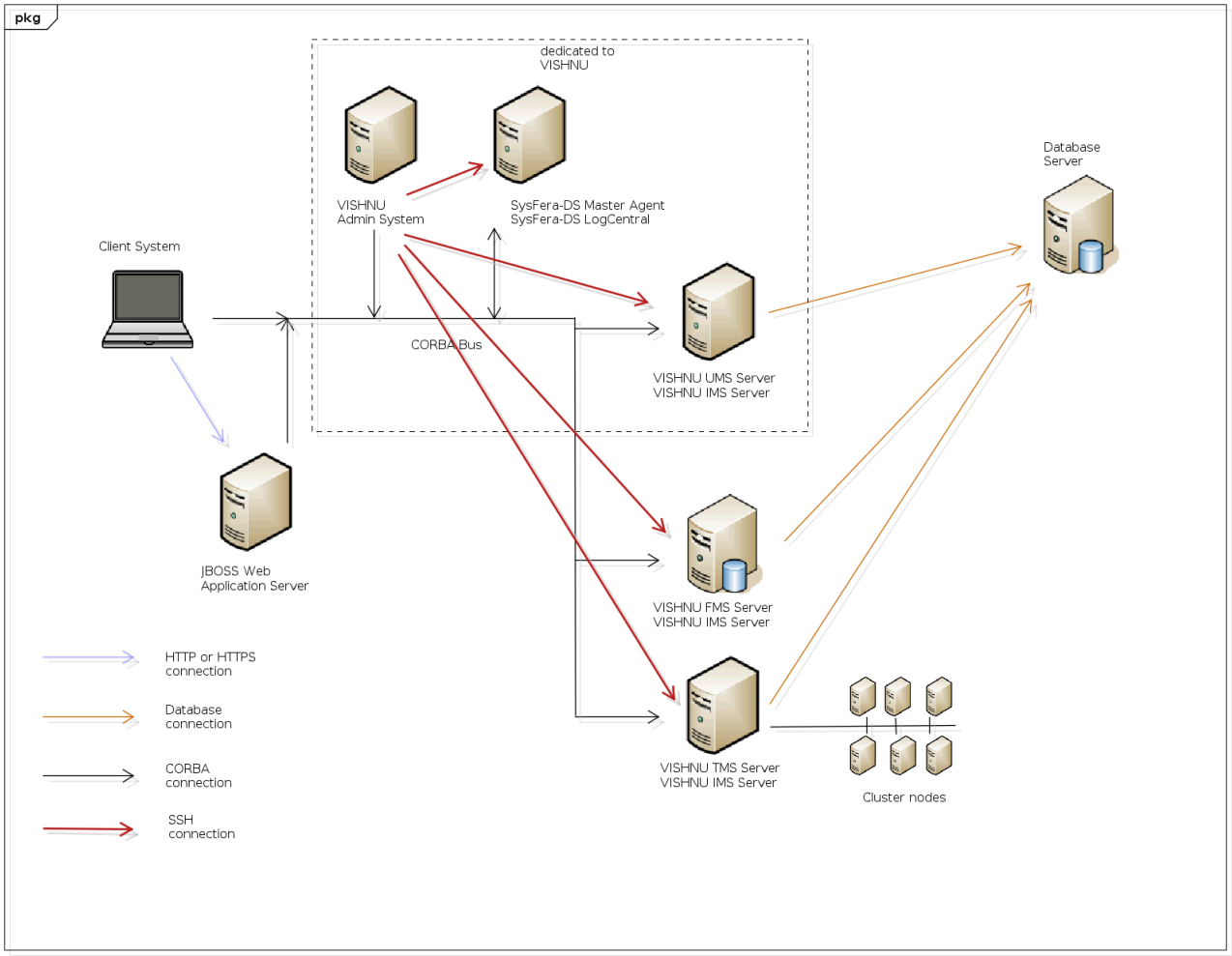

Figure 2.3: IMS deployment overview

2.3.2 IMS deployment diagram

This diagram shows the generic deployment pattern of VISHNU processes. All IMS SeD processes must connect to the same VISHNU database. The TMS and FMS SeD are optional and if present they can be used to fulfill some IMS services (e.g. load shedding).

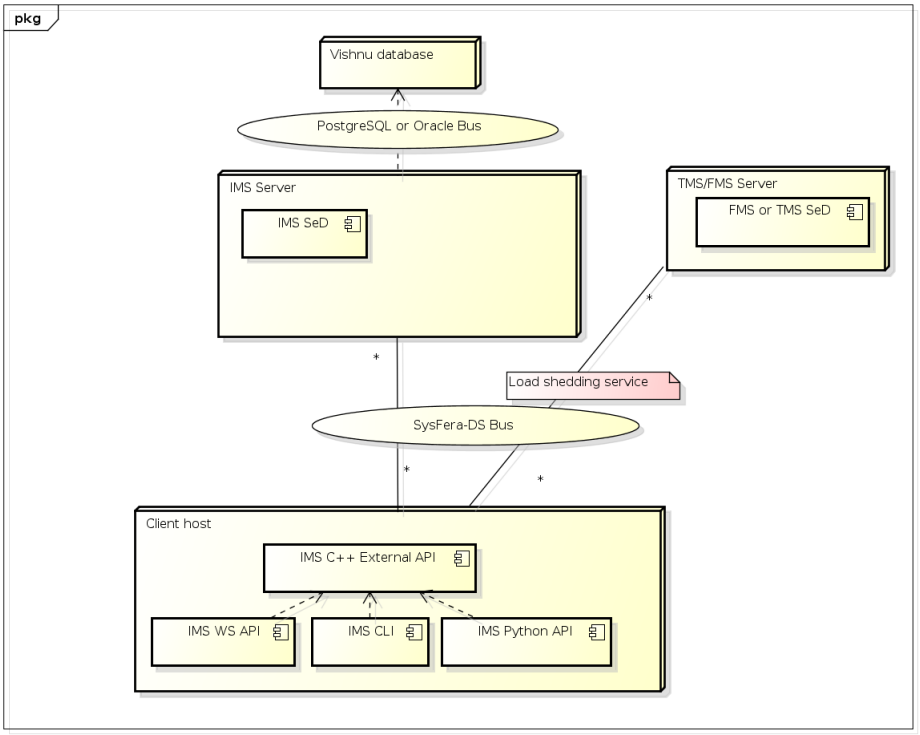

Figure 2.4: IMS deployment diagram

2.3.3 SysFera-DS Bus Details

This diagram shows the communication paths between the Client host and an IMS SeD using the SysFera-DS Bus. The SysFera-DS MasterAgent is a SysFera-DS agent that can be executed on a dedicated host or on the same host as the IMS SeD. All the communications between the entities here are done using the CORBA IIOP (Internet Inter-ORB) protocol and the communications can be tunneled through SSH tunnels if necessary. The MasterAgent entity is involved in the choice of one IMS SeD in the case of several available IMS SeD. The choice will be transparent to the user as all IMS SeD connect to the same database. The diagram shows here the communication paths. The IMS client can be client to the FMS or TMS SeD so they are presented on the diagram. Nevertheless, this call may only happen if a specific call to the load shed service happens on the machine where the FMS and/or TMS SeD are running.

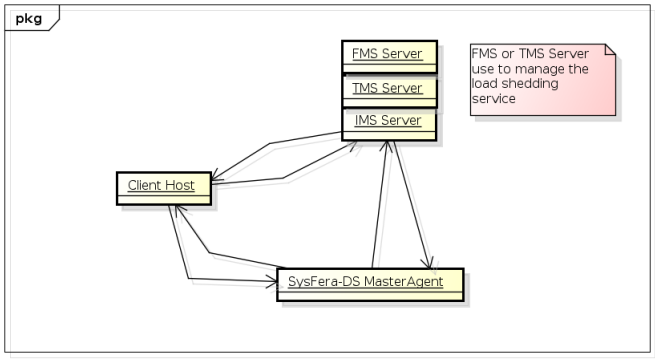

Figure 2.5: SysFera-DS Bus Details

Chapter 3

Overview of the IMS modelization

3.1 Functional view

The following diagram is functional. The IMS client can ask the IMS SeD to fill four functions. It is important to note that the IMS client can access specific TMS and FMS services. Morevover, all the SeD will have to save the command that was used to launch them.

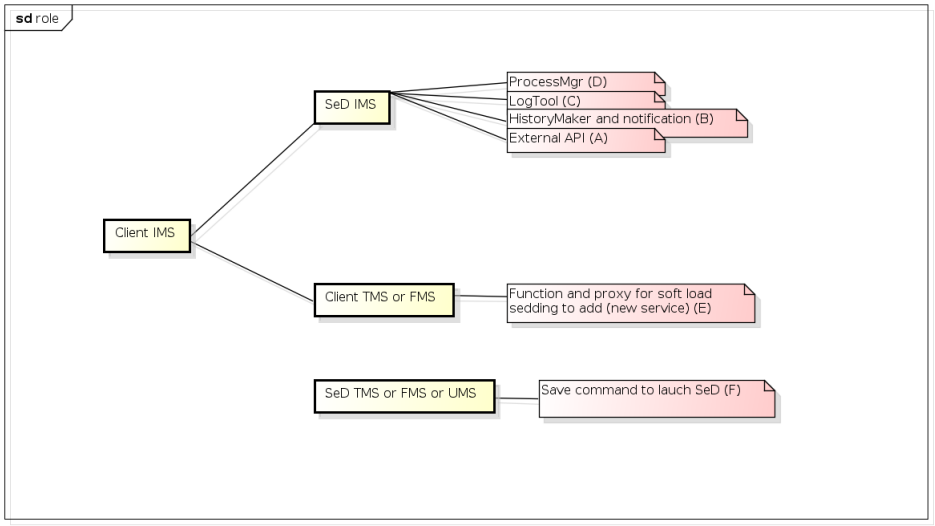

Figure 3.1: Functional roles

The meaning of the roles (using the references)

- A: The external API corresponds to the functionnalities that are to be called from the IMS client
- B: The history maker and notification corresponds to the function of getting the state of the machine and saving it in the database automatically. Moreover it notifies the admin if the state has reached a threshold for the machine.
- C: LogTool corresponds to the part of the SeD that will be linked to the log central in the publisher/subscriber way and that gets the information of a new sed or a disconnexion on the machine the IMS SeD is over.
- D: Process Manager is responsible for relaunching a dead SeD, stopping a SeD or making the load shedding.
- E: Client TMS or FMS that will have new functionalities to call the TMS/FMS SeD
- F: Modification of the SeD to hava them save their parameters when launched, it enables to restart the SeD with the same parameters.

3.2 Roles description

The IMS module can play eight different roles. For each role, there is a reference to the above functional figure with the letter corresponding to the functional element.

- The hard load shedding : It means the death of all the VISHNU processes on a machine. These processes will not be automatically relauched. Only the GoDiet tool, made to launch the processes can make it. Reference: D.
- The soft load shedding : It means the end of the current jobs (submitted using TMS) and file transfers (made with FMS) on a machine. The stopped commands are set to fail in the database. Reference: E+D
- The export of the commands in a format : It generates a shell script containing all the recorded commands made during a session. Commands such as connect or change password cannot be exported. Because the server does not know the user password, the script cannot be automatically executed, a connect call must be added. Reference: A
- The live monitor : It means getting the current state of the machines. Reference: A
- The delayed monitor : It means getting past states of a machine and automatically record the states of the machine with the time passing. Reference: A+B
- The notification monitor : It means to notify automatically the administrator of an abnormal behaviour of the VISHNU system. Reference: B
- The automatic restart : Once a SeD is down, the IMS SeD is informed of it and it tries to relaunch it. If an agent is down, it cannot be the IMS SeD that restart it automatically but GoDIET will make it. Reference: A+C
- The manual restart : It can be made using GoDIET, but it is also available throught the IMS API. Only VISHNU SeD can be relaunched throught the API. Reference: F+D

3.3 Classes and roles

The following figure presents the links between the functional roles and the classes corresponding. There are two main groups of classes. The DATA, that correspond to the external API and that deal with the database, and the CONTROLER that only interact with the DATA classes

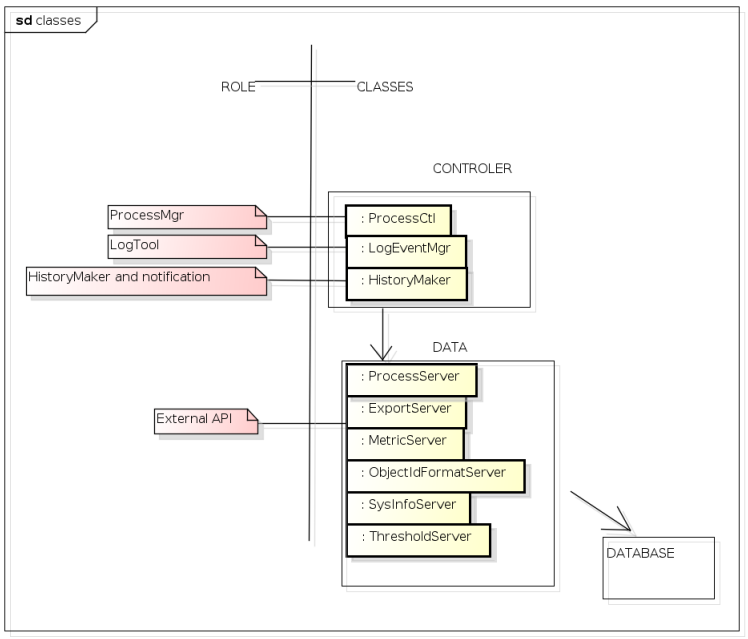

Figure 3.2: Classes implementing the roles

Chapter 4

Internal API specification

4.1 Generic definition formats presentation

This section presents the formats used in this chapter to describe the services provided by the internal API.

4.1.1 Service definition format

Access

Here is detailed the access level of the service 'myService' (i.e. the privilege required to use it)

Parameters

The following table contains all the input and output parameters of the service, along with their type and description.

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	This is an example of a required string input parameter	IN
listOfJobs	string	ListJobs	This is an example of an object output parameter that is serialized as a string	OUT

Description

Here is detailed the purpose of the service 'myService'

Return Value

Here are detailed the different return codes provided by the service.

Name	Description
VISHNU_OK	The service has been performed successfully.
TMS_UNKNOWN_MACHINE	This is the human-readable generic message that will be available to the user of the API.

Used by this(these) API function(s):

This shows the list of functions from the external Vishnu API (see [D1_1c]) that use this service.

4.2 Definition of the services of the package

4.2.1 Service int_exportCommands

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
oldSessionId	string	n/a	The id of the session to export (session has ended)	IN
filename	string	n/a	The path of the output file containing the Vishnu shell commands	INOUT
options	string	ExportOp	Options for the export	IN

Description

The int_exportCommands() function exports all the commands made by a user during a session

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.2 Service int_getMetricCurrentValue

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
machineId	string	n/a	The id of the machine	IN
metricValue	string	Metric	Value of the metric	OUT
options	string	CurMetricOp	Options	IN

Description

The int_getMetricCurrentValue() function retrieve the current value of a metric on a machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.3 Service int_getMetricHistory

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
machineId	string	n/a	The id of the machine	IN
metricType	string	MetricType	Type of metric	IN
metricValues	string	ListMetric	List of metric values	OUT
endTime	string	MetricHistOp	End time of metric history	IN

Description

The int_getMetricHistory() function retrieve the history of values of a metric on a machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.4 Service int_getProcesses

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
process	string	ListProcesses	The list of the Vishnu processes on the machine	OUT
options	string	ProcessOp	The id of the machine the user wants the running processes	IN

Description

The int_getProcesses() function gets the list of the processes running over a front machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.5 Service int_setSystemInfo

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
systemInfo	string	SystemInfo	Contains system information to store in Vishnu database	IN

Description

The int_setSystemInfo() function updates the system information of a machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.6 Service int_setSystemThreshold

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
threshold	string	Threshold	The type of the metric to set	IN

Description

The int_setSystemThreshold() function sets a threshold on a machine of a system

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.7 Service int_getSystemThreshold

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
options	string	ThresholdOp	The options for the threshold	IN
value	string	ListThreshold	The threshold values	OUT

Description

The int_getSystemThreshold() function gets a System threshold on a machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.8 Service int_defineUserIdentifier

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
format	string	n/a	The new format to use	IN

Description

The int_defineUserIdentifier() function defines the shape of the identifiers automatically generated for the users

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.9 Service int_defineMachineIdentifier

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
format	string	n/a	The new format to use	IN

Description

The int_defineMachineIdentifier() function defines the shape of the identifiers automatically generated for the machines

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.10 Service int_defineJobIdentifier

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
format	string	n/a	The new format to use	IN

Description

The int_defineJobIdentifier() function defines the shape of the identifiers automatically generated for the jobs

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.11 Service int_defineTransferIdentifier

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
format	string	n/a	The new format to use	IN

Description

The int_defineTransferIdentifier() function defines the shape of the identifiers automatically generated for the file transfers

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.12 Service int_loadShed

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
machineId	string	n/a	The id of the machine to stop	IN
loadShedType	string	LoadShedType	Selects a load shedding mode (SOFT: stops all services and they can be restarted, HARD: stops all services, they cannot be restarted)	IN

Description

The int_loadShed() function load sheds a machine

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.13 Service int_setUpdateFrequency

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
freq	int	n/a	Frequency the data are updated, in second	IN

Description

The int_setUpdateFrequency() function sets the update frequency of the IMS tables

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.14 Service int_getUpdateFrequency

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
freq	int	n/a	Frequency the data are updated, in second	OUT

Description

The int_getUpdateFrequency() function gets the update frequency of the IMS database

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.15 Service int_restart

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
machineId	string	n/a	The id of the machine where the component is to be relaunched	IN
type	string	RestartType	The type of restart	IN
options	string	RestartOp	The options to restart	IN

Description

The int_restart() function restarts the whole VISHNU infrastructure. Actions are saved and restarted from the beginning once the infrastructure has been restarted

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.16 Service int_stop

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN

Parameter	Type	Serialized type	Description	Mode
process	string	Process	The process to stop	IN

Description

The int_stop() function to stop a sed

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

4.2.17 Service int_getSystemInfo

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The session key	IN
res	string	ListSysInfo	The set of system information	OUT
options	string	SysInfoOp	Options for the sysinfo	IN

Description

The int_getSystemInfo() function to get system info

Return Value

An error code is returned when an error occurs during the execution of the service

Name	Description

Used by this(these) API function(s):

None

Chapter 5

Internal class and data structures

5.1 Introduction

This chapter introduces the details of the implementation of the different components described in chapter 2. It is composed of three sections:

- **Client modelization**: describes the classes used to implement the *IMS Client* component.
- **Server modelization**: describes the classes used to implement the *IMS SeD* component.
- **Data modelization**: describes the data structure used to implement the *IMS Client* component and the *IMS SeD* component.

5.2 IMS Proxy ClassDiagram

The following figure presents the class diagram on the client side

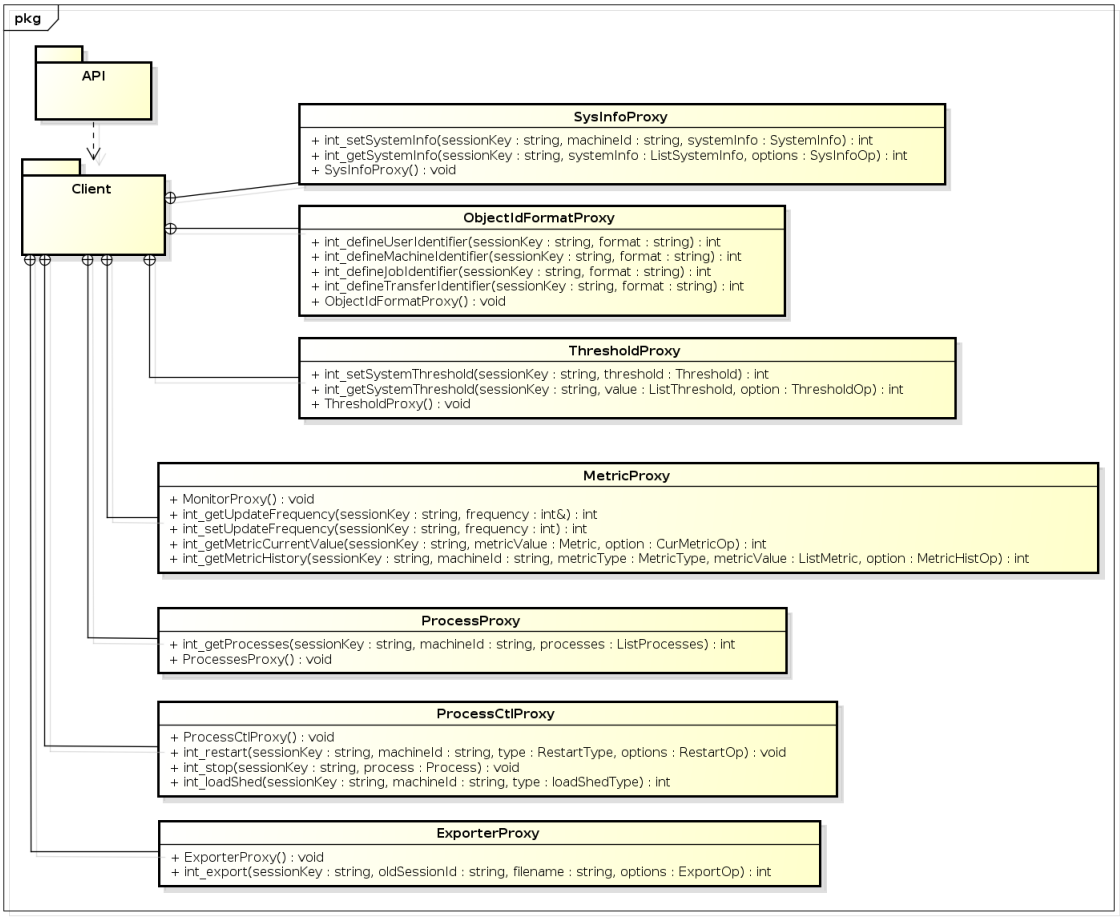

Figure 5.1: IMS Proxy ClassDiagram

5.3 IMS Server ClassDiagram

The following figure presents the class diagram on the server side

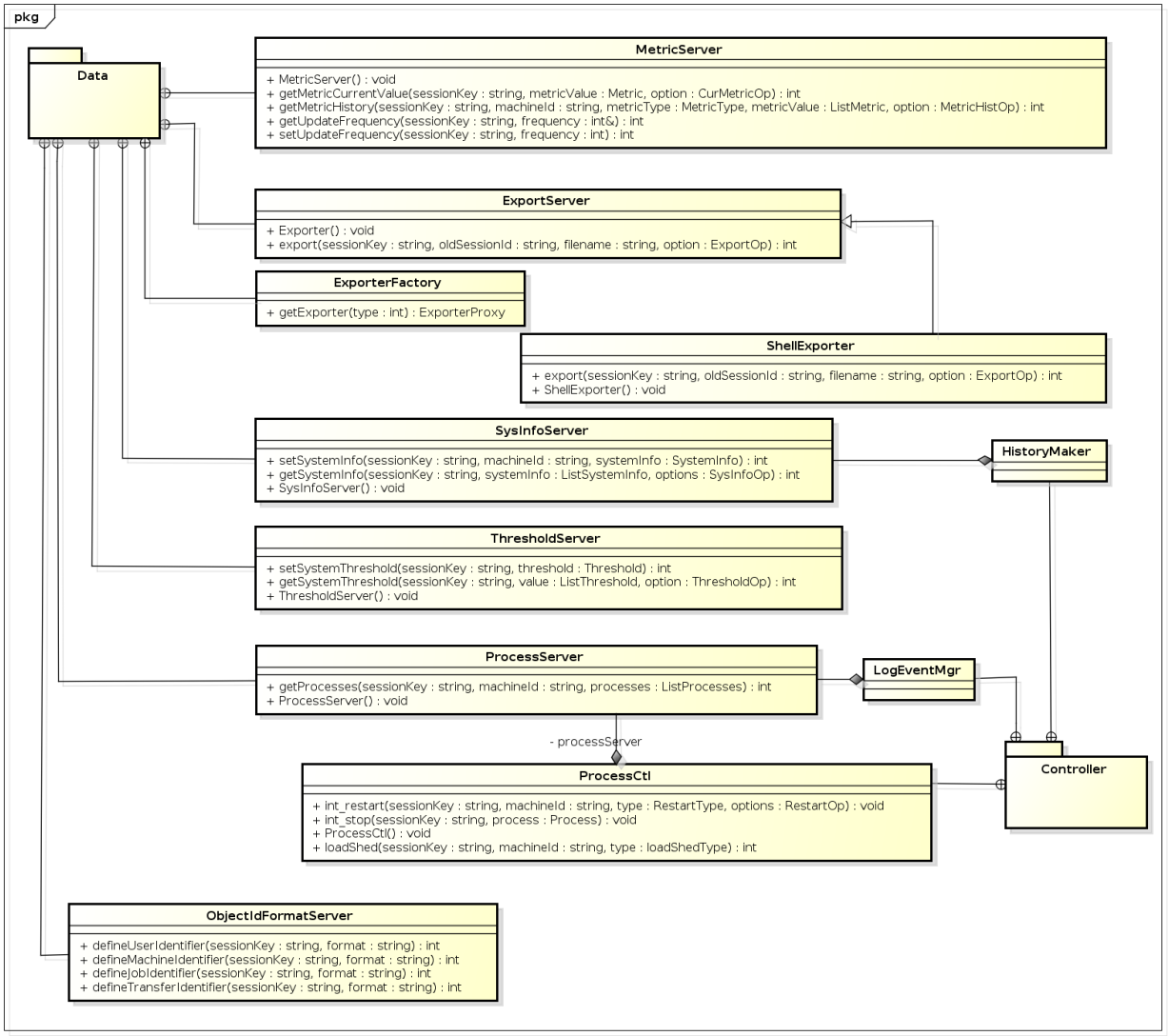

Figure 5.2: IMS Server ClassDiagram

5.4 IMS Datatype ClassDiagram

The following figure presents the links between the datatypes

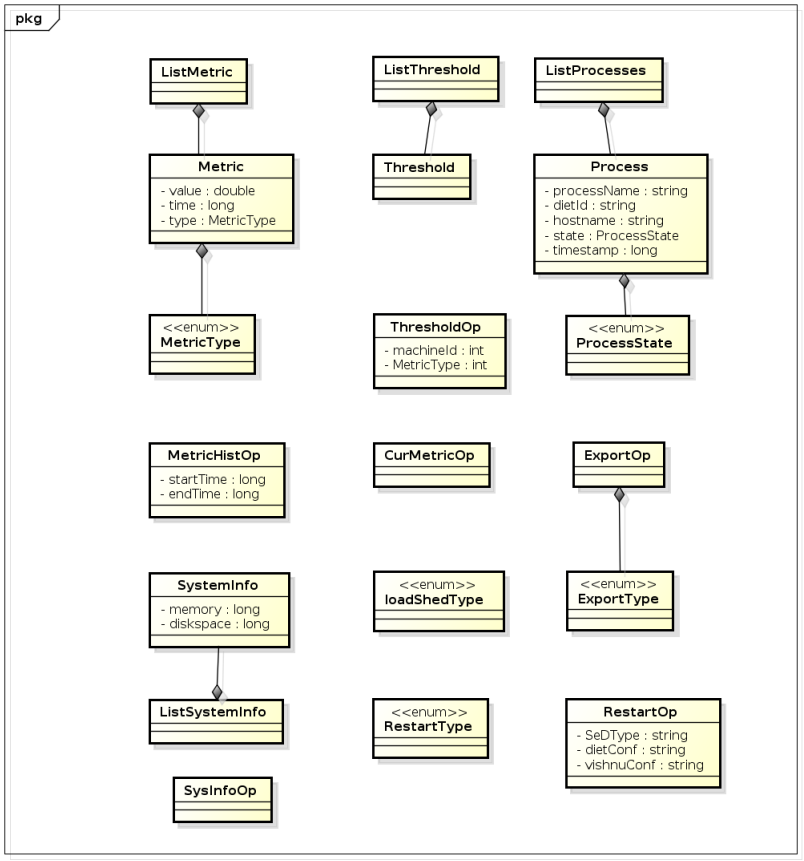

Figure 5.3: IMS Datatype ClassDiagram