

D4.1a - VISHNU Information Management System Package Design

| |
|----------------------|
| COLLABORATORS |
|----------------------|

| | | | |
|---------------|--|---------------|------------------|
| | <i>TITLE :</i> D4.1a - VISHNU Information Management System Package Design | | |
| <i>ACTION</i> | <i>NAME</i> | <i>DATE</i> | <i>SIGNATURE</i> |
| WRITTEN BY | Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé | April 1, 2011 | |

| |
|-------------------------|
| REVISION HISTORY |
|-------------------------|

| | | | |
|--------|------------|---------------------|---------|
| NUMBER | DATE | DESCRIPTION | NAME |
| 1 | 29/03/2011 | Deliverable version | SysFera |

Contents

| | | |
|----------|---|-----------|
| 1 | Document presentation | 1 |
| 1.1 | Document objectives | 1 |
| 1.2 | Document structure | 1 |
| 1.3 | References | 1 |
| 1.4 | Acronyms | 1 |
| 1.5 | Glossary | 2 |
| 2 | System Architecture | 3 |
| 2.1 | Overview of the IMS software infrastructure | 3 |
| 2.2 | Deployment aspects of IMS | 3 |
| 2.3 | Architecture diagrams | 4 |
| 2.3.1 | IMS Deployment Diagram | 4 |
| 2.3.2 | IMS client-side components | 4 |
| 2.3.3 | IMS server-side components | 5 |
| 2.3.4 | SysFera-DS Bus Details | 6 |
| 3 | Overview of the internal IMS organisation | 8 |
| 3.1 | The role modelisation | 8 |
| 3.1.1 | Architecture diagrams | 8 |
| 3.1.1.1 | role | 8 |
| 3.2 | The organisation | 8 |
| 3.3 | The role modelisation | 9 |
| 3.3.1 | Architecture diagrams | 9 |
| 3.3.1.1 | classes | 9 |
| 4 | Internal API specification | 10 |
| 4.1 | Generic definition formats presentation | 10 |
| 4.1.1 | Service definition format | 10 |
| 4.2 | Definition of the services of the package | 11 |
| 4.2.1 | Service int_exportCommands | 11 |
| 4.2.2 | Service int_getMetricCurrentValue | 11 |

| | | |
|----------|--|-----------|
| 4.2.3 | Service int_getMetricHistory | 12 |
| 4.2.4 | Service int_getProcesses | 12 |
| 4.2.5 | Service int_setSystemInfo | 13 |
| 4.2.6 | Service int_setSystemThreshold | 13 |
| 4.2.7 | Service int_getSystemThreshold | 13 |
| 4.2.8 | Service int_defineUserIdentifier | 14 |
| 4.2.9 | Service int_defineMachineIdentifier | 14 |
| 4.2.10 | Service int_defineJobIdentifier | 15 |
| 4.2.11 | Service int_defineTransferIdentifier | 15 |
| 4.2.12 | Service int_loadShed | 16 |
| 4.2.13 | Service int_setUpdateFrequency | 16 |
| 4.2.14 | Service int_getUpdateFrequency | 17 |
| 5 | Internal class and data structures | 18 |
| 5.1 | Introduction | 18 |
| 5.2 | IMS client modelization | 18 |
| 5.2.1 | Class diagrams | 18 |
| 5.2.1.1 | IMS Proxy ClassDiagram | 18 |
| 5.3 | IMS SeD modelization | 19 |
| 5.3.1 | Class diagrams | 19 |
| 5.3.1.1 | IMS Server ClassDiagram | 19 |
| 5.4 | IMS data modelization | 20 |
| 5.4.1 | Class diagrams | 20 |
| 5.4.1.1 | IMS Datatype ClassDiagram | 20 |

List of Figures

| | | |
|-----|--------------------------------------|----|
| 2.1 | IMS Deployment Diagram | 4 |
| 2.2 | IMS client-side components | 5 |
| 2.3 | IMS server-side components | 6 |
| 2.4 | SysFera-DS Bus Details | 7 |
| 3.1 | role | 8 |
| 3.2 | classes | 9 |
| 5.1 | IMS Proxy ClassDiagram | 19 |
| 5.2 | IMS Server ClassDiagram | 20 |
| 5.3 | IMS Datatype ClassDiagram | 20 |

Chapter 1

Document presentation

1.1 Document objectives

This document presents the detailed internal design of the Information Management Service (IMS) package. The purpose of this package is to handle all aspects of information management within the VISHNU system. The functional and non-functional requirements for this package 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 high-level overview of the system architecture.
- Chapter 3 contains an overview of the internal IMS organisation.
- 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

1.4 Acronyms

- **API**: Application programming interface
 - **CLI**: Command line interface
 - **DB**: DataBase
 - **n/a**: Not Applicable (used for serializable capability in function descriptions)
-

- **SeD:** A Server Daemon is a SysFera-DS agent that provides services through the SysFera-DS API.
- **IMS:** Information management service
- **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.
 - **Serialized type:** this is a class of data (C++ Class) which instances can be serialized in a XML string before being sent over an API (to or from the API). The data is deserialized on the other side of the channel in order to re-build the same instance of the class.
 - **SysFera-DS:** open-source middleware developped by SysFera.
-

Chapter 2

System Architecture

2.1 Overview of the IMS software infrastructure

We present in this section a detailed description of the IMS package architecture 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 server package has been split into eight different interrelated components. The diagrams shown in section 2.3 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 middle layer of the server side. The services announced previously are performed here by combining a set of classes defined in the two following components.
- **IMS Client** contains intermediate (proxy) classes providing remote access to the business objects of **IMS SeD**.
- **IMS SeD** contains all classes implementing business objects by encapsulating the processing provided through the internal API. There must be an IMS SeD on each machine to monitor. A machine without IMS SeD cannot be monitored.
- **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 manipulated by the IMS SeD.

2.2 Deployment aspects of IMS

We explain here how the IMS package will be deployed in a physical hardware as illustrated in figure 2.1 where each cube represents an environnement in which a component or a set of components execute. The IMS consists of:

- **IMS SeD** is the provider of all IMS services. It consists of the IMS SeD component which gathers all IMS services published. There is one IMS SeD per machine to supervise.
 - **Client host** is IMS service requester. It contains all components allowing to make a IMS service request.
 - **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 Architecture diagrams

2.3.1 IMS Deployment Diagram

This diagram shows the classes of entities that must be deployed for the VISHNU IMS application to work. All IMS SeD entities should connect to the same VISHNU database. Moreover, the TMS and FMS SeD are not there because they are necessary, but to show that the IMS client can communicate with them if they are present.

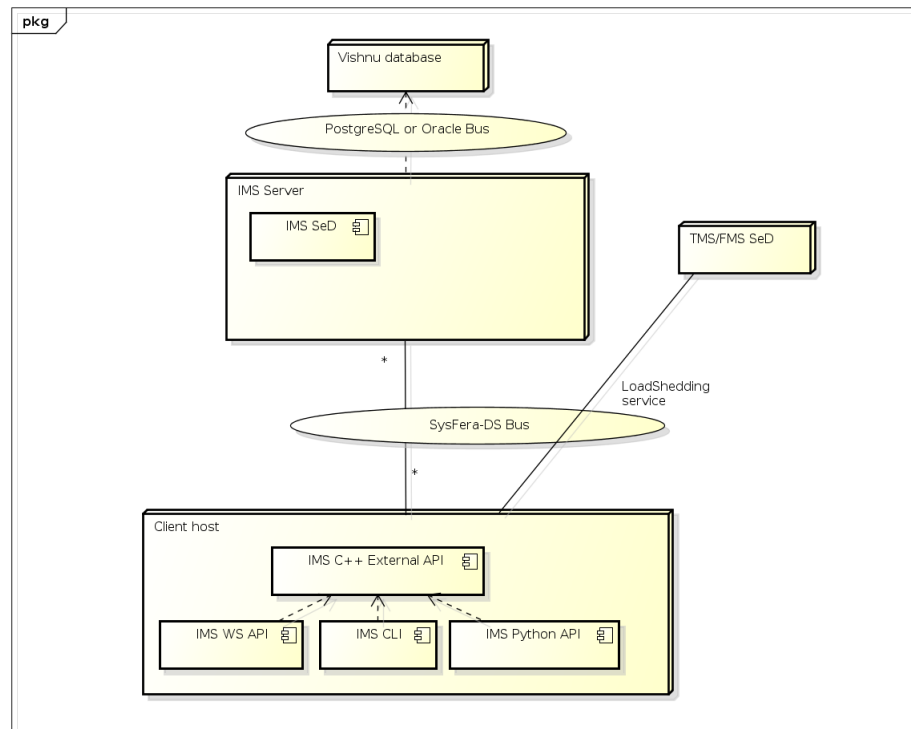


Figure 2.1: IMS Deployment Diagram

2.3.2 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++).

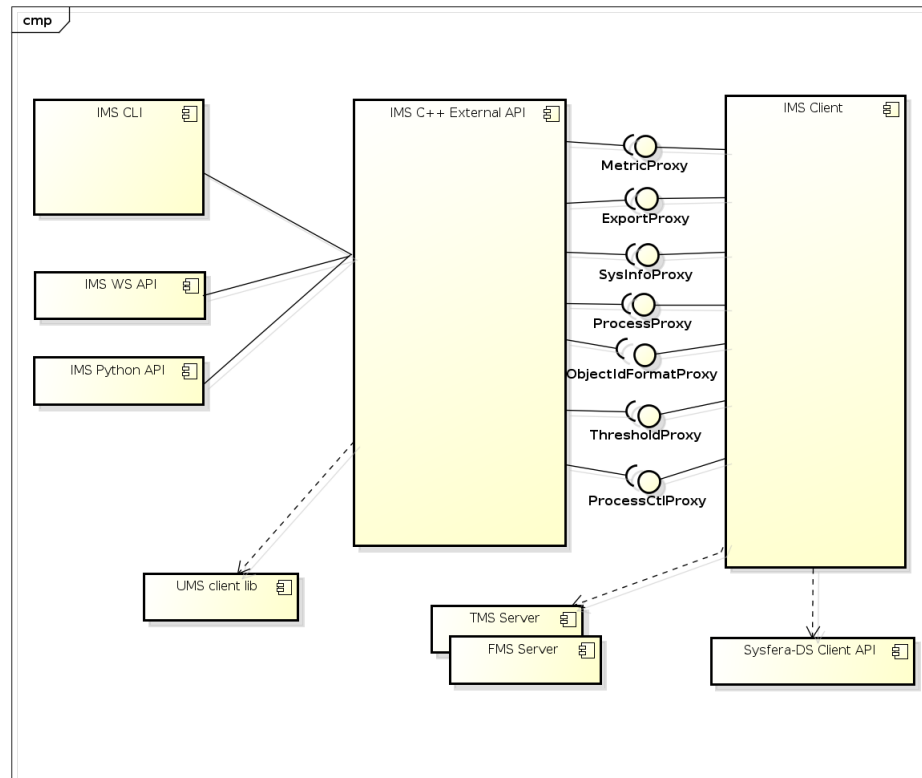


Figure 2.2: IMS client-side components

2.3.3 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.

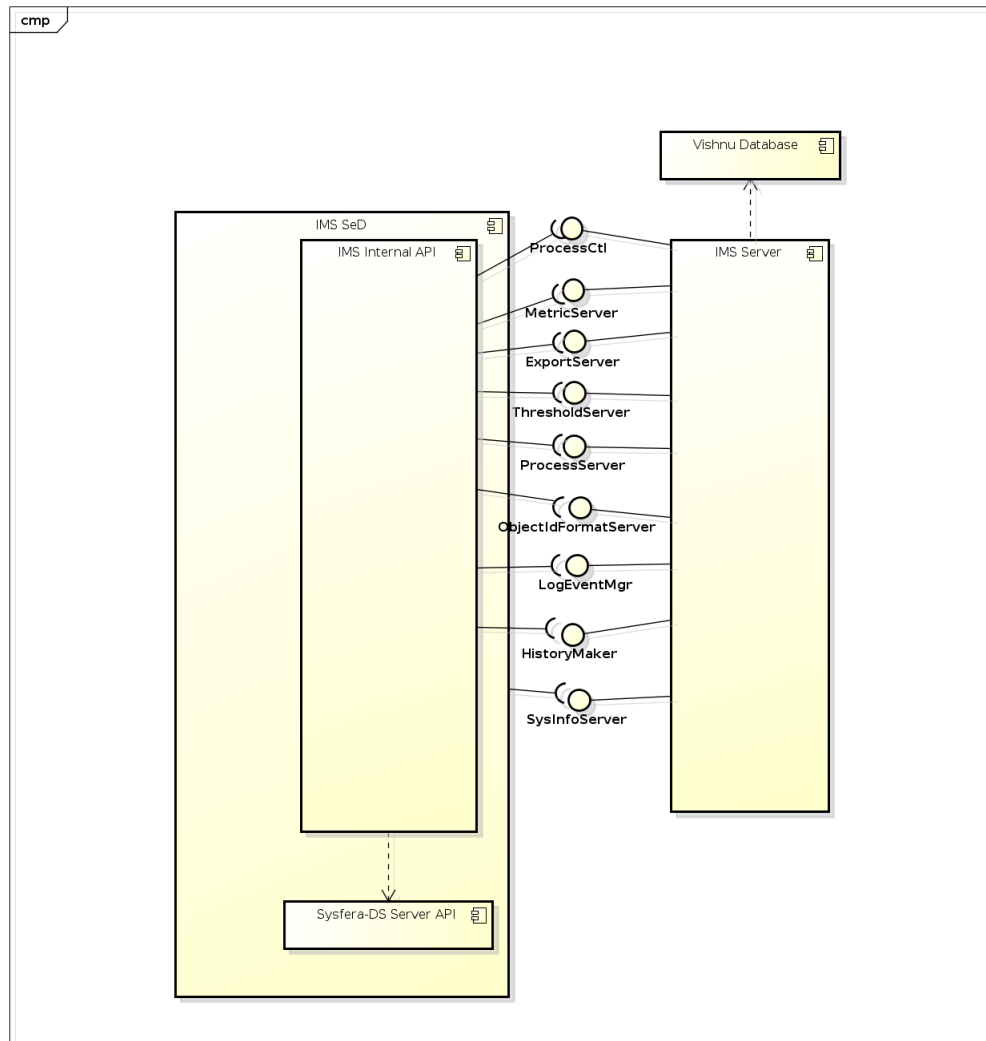


Figure 2.3: IMS server-side components

2.3.4 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 to load shed the machine where the FMS or TMS SeD are.

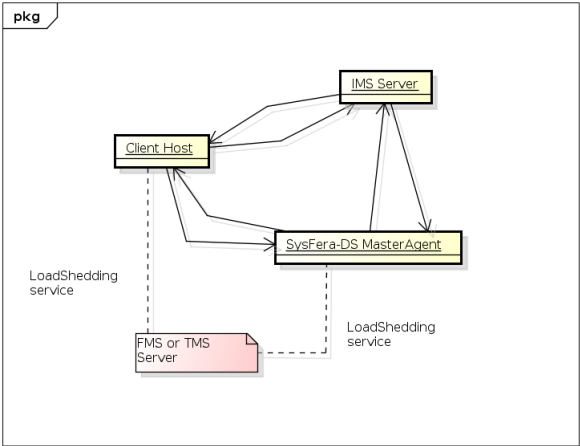


Figure 2.4: SysFera-DS Bus Details

Chapter 3

Overview of the internal IMS organisation

3.1 The role modelisation

3.1.1 Architecture diagrams

3.1.1.1 role

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. Moreover, all the SeD will have to save the command that was used to launch them.

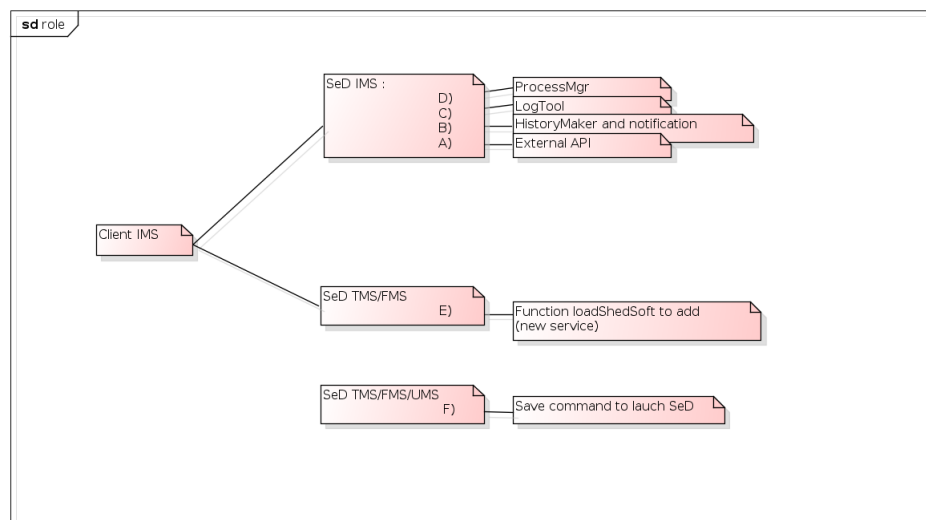


Figure 3.1: role

3.2 The organisation

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 relaunched. 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. Reference: F+D

3.3 The role modelisation

3.3.1 Architecture diagrams

3.3.1.1 classes

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

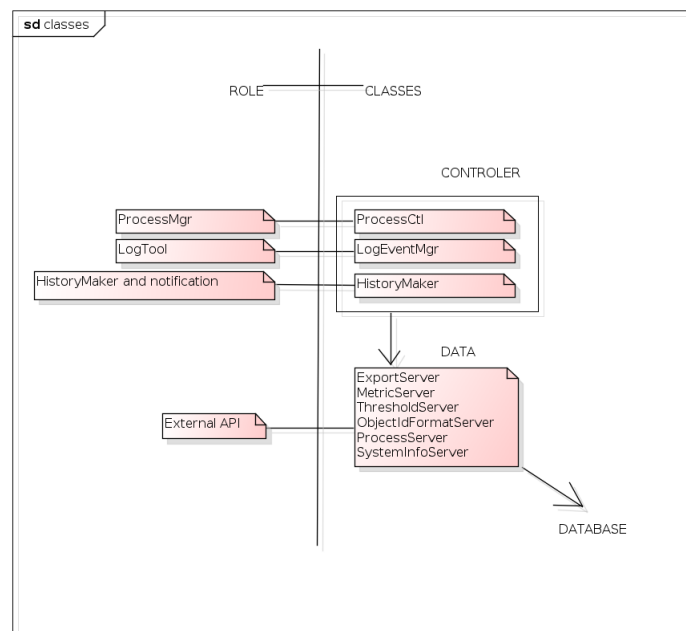


Figure 3.2: classes

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 |

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 |
| metricType | string | MetricType | Type of metric | IN |
| metricValue | string | Metric | Value of the metric | OUT |

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 |
| startTime | long | n/a | Start time of metric history | IN |
| endTime | long | n/a | End time of metric history | IN |
| metricType | string | MetricType | Type of metric | IN |
| metricValues | string | ListMetric | List of metric values | OUT |

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 |
| machineId | string | n/a | The id of the machine the user wants the running processes | IN |
| process | string | ListProcesses | The list of the Vishnu processes on the machine | OUT |

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 |
| machineId | string | n/a | The id of the machine | 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 |
| machineId | string | n/a | The id of the machine | IN |
| thresholdType | string | MetricType | The type of the metric to set | IN |
| value | double | n/a | The threshold value | 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 |
| machineId | string | n/a | The id of the machine | IN |
| type | string | MetricType | The threshold type desired | IN |
| value | double | n/a | The threshold value | 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 |
|-------------------------|--------|-----------------|-----------------------|------|
| <code>sessionKey</code> | string | n/a | The session key | IN |
| <code>format</code> | 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 |
|-------------------------|--------|-----------------|-----------------------|------|
| <code>sessionKey</code> | string | n/a | The session key | IN |
| <code>format</code> | 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

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 (Architecture). 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 client modelization

5.2.1 Class diagrams

5.2.1.1 IMS Proxy ClassDiagram

The following figure presents the class diagram on the client side

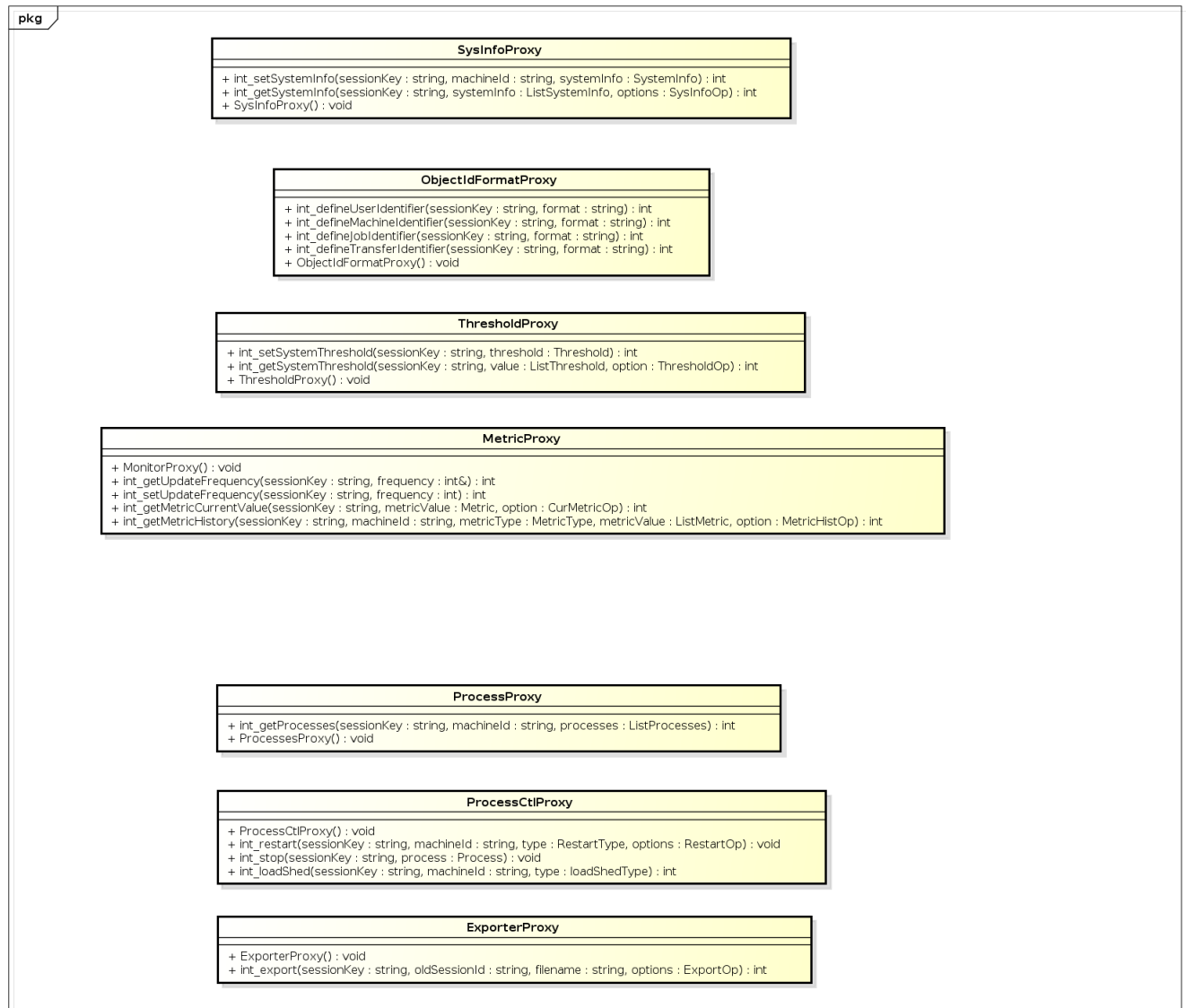


Figure 5.1: IMS Proxy ClassDiagram

5.3 IMS SeD modelization

5.3.1 Class diagrams

5.3.1.1 IMS Server ClassDiagram

The following figure presents the class diagram on the server side

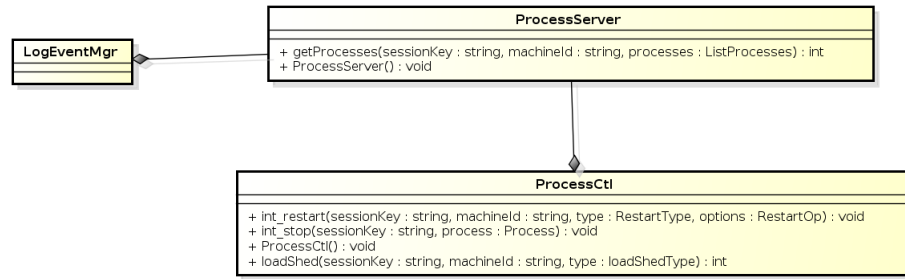


Figure 5.2: IMS Server ClassDiagram

5.4 IMS data modelization

5.4.1 Class diagrams

5.4.1.1 IMS Datatype ClassDiagram

The following figure presents the links between the datatypes

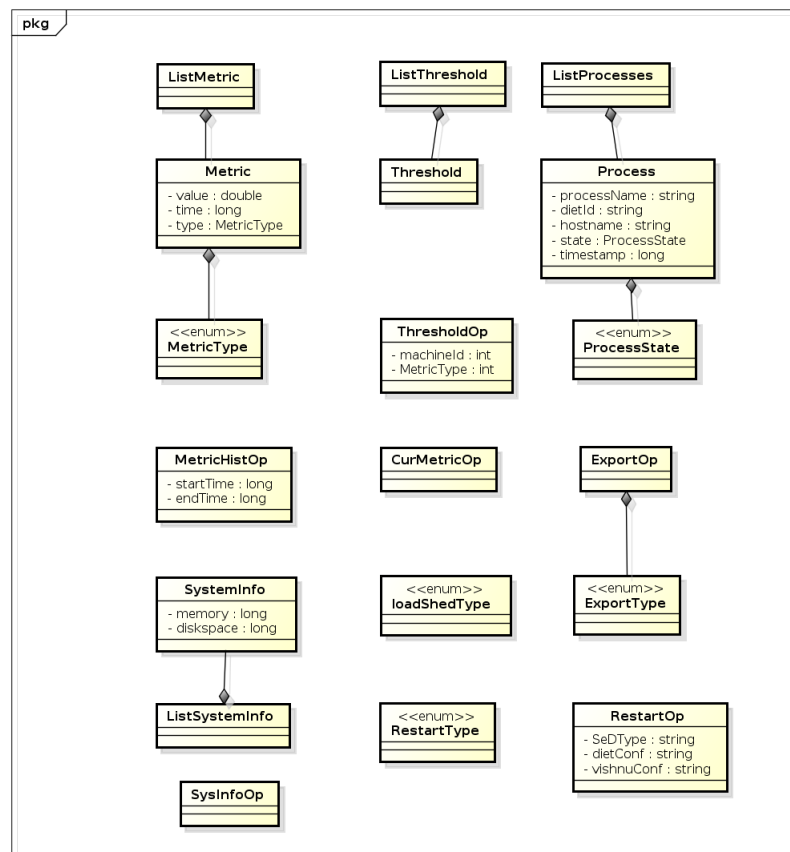


Figure 5.3: IMS Datatype ClassDiagram