D2.1a - VISHNU User Management System Package Design

COLLABORATORS			
	TITLE : D2.1a - VISHNU User Management System Package Design		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé	December 16, 2011	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1	25/01/2011	Deliverable version	SysFera
2	24/02/2011	Modified addUser and addMachine signature.	SysFera
3	16/12/2011	Replace all Oracle occurrences by MySQL	SysFera

Contents

List of Figures

Chapter 1

Document presentation

1.1 Document objectives

This document presents the detailed internal design of the Users Management System (UMS) package. The purpose of this package is to handle all aspects of user management and session 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 describes the internal API used for remote procedure calls through SysFera-DS.
- Chapter 4 describes the internal class 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 Appliable (used for serializable capability in function descriptions)
- **SeD**: A Server Daemon is a SysFera-DS agent that provides services through the SysFera-DS API.
- **UMS**: Users management system
- **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 UMS software infrastructure

We present in this section a detailed description of the UMS 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 UMS 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.
- **UMS Client** contains intermediate (proxy) classes providing remote access to the business objects of **UMS Server**.
- **UMS Server** contains all classes implementing business objects by encapsulating the processing provided through the internal API.
- **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.
- **UMS Monitor Daemon** which the only role is to keep an eye on the session inactivity by checking the TIME_OUT parameter in the **Vishnu Database.**
- **Vishnu Database** stores all data manipulated by the UMS Server.

2.2 Deployment aspects of UMS

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

- **Main UMS Server** is the provider of all UMS services. It consists of the UMS Monitor component and what we called the UMS SeD (UMS Server daemon) which gathers all UMS services published.
- **Secondary UMS Server** is optional and contains only the UMS SeD allowing to make a UMS service request.
- **Client host** is UMS service requester. It contains all components allowing to make a UMS 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 MySQL or PostgreSQL database.

It is important to note that we can have several Secondary UMS servers (where an UMS Sed is running in each) but only one instance of UMS Monitor daemon running in the Main UMS Server as we can see in figure 2.1.

2.3 Architecture diagrams

2.3.1 UMS Deployment Diagram

This diagram shows the classes of entities that must be deployed for the Vishnu UMS application to work. One Main UMS Server contains both a UMS SeD component and a UMS Monitor Daemon component. The Secondary UMS Server entity is optional and can be duplicated to improve performance and robustness. All UMS Server entities connect to the same Vishnu database. It is important to note that the different classes are here shown on different nodes but they can also be deployed on the same node.

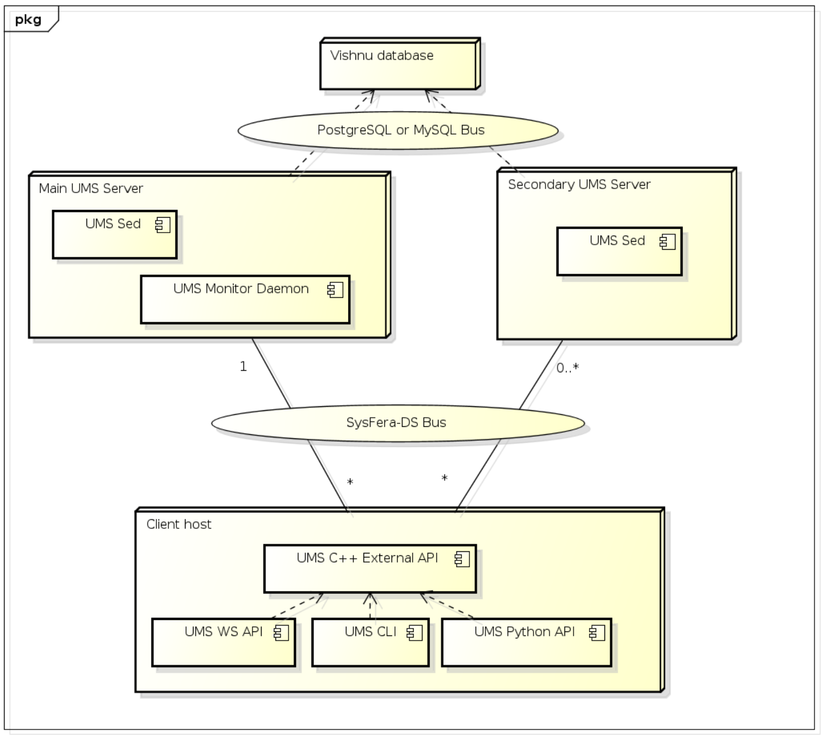

Figure 2.1: UMS Deployment Diagram

2.3.2 UMS client-side components

This diagram shows the components that compose the client side of the Vishnu UMS system and their interfaces. Two services among all the services of the UMS external API (see ref. D1.1c) are shown here for example. These services are consumed by several user interfaces: command-line, web services and Python. All the interfaces of the UMS Client component are shown.

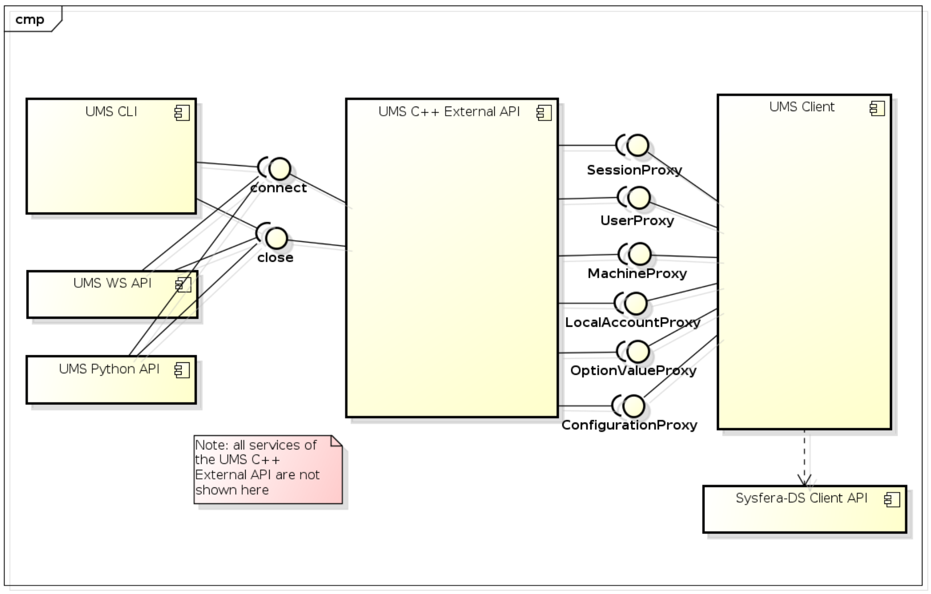

Figure 2.2: UMS client-side components

2.3.3 UMS server-side components

This diagram shows the components that compose the server side of the Vishnu UMS system and their interfaces. Two services among all the services of the UMS internal API are shown here for example. These services are consumed by the UMS Client component through the SysFera-DS API. All the interfaces of the UMS Server component are shown.

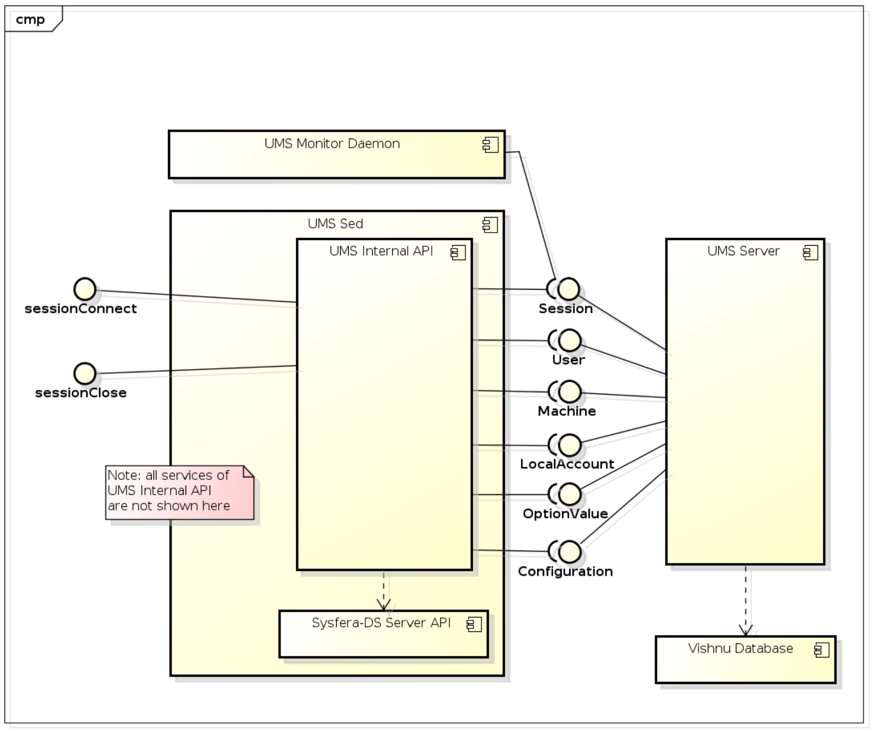

Figure 2.3: UMS server-side components

2.3.4 SysFera-DS Bus Details

This diagram shows the communication paths between the Client host and the UMS Main and Secondary server 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 UMS Server. 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 UMS Server in the case of several available UMS servers. The choice will be transparent to the user as all UMS Servers connect to the same database. The diagram shows here all the communication paths in the case where the Main UMS Server is chosen by the MasterAgent.

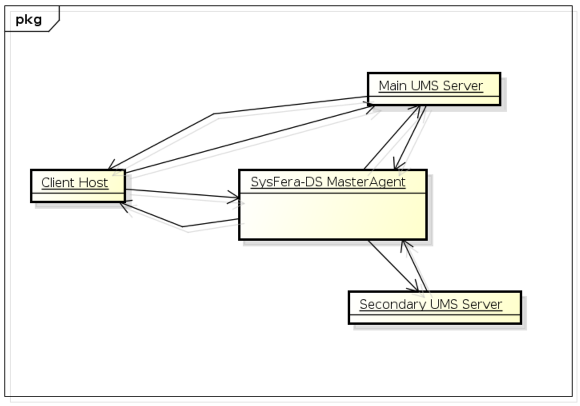

Figure 2.4: SysFera-DS Bus Details

Chapter 3

Internal API specification

3.1 Generic definition formats presentation

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

3.1.1 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.

3.2 Definition of the services of the package

3.2.1 Service commandList

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
options	string	ListCmdOptions	allows the user to list commands by using several optional criteria: a period, specific session and for admin to list all commands of all VISHNU users or commands from a specific user	IN
listCommands	string	ListCommands	listCommands is the list of commands	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The commandList() function lists the commands

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):

UMS::listHistoryCmd

3.2.2 Service sessionConnect

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
userId	string	n/a	userId represents the VISHNU user identifier	IN
password	string	n/a	password represents the password of the user	IN
clientKey	string	n/a	The SSH key that identifies the client host	IN
clientHostname	string	n/a	The full DNS name of the client host	IN
options	string	ConnectOptions	options is an object which encapsulates the options available for the connect method. It allows the user to choose the way for closing the session automatically on TIMEOUT or on DISCONNECT and the possibility for an admin to open a session as if he/she was a specific user	IN
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The sessionConnect() function opens 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):

UMS::connect

3.2.3 Service sessionReconnect

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
userId	string	n/a	userId represents the VISHNU user identifier	IN
password	string	n/a	password represents the password of the user	IN
clientKey	string	n/a	The SSH key that identifies the client host	IN
clientHostname	string	n/a	The full DNS name of the client host	IN
sessionId	string	n/a	sessionId is the identifier of the session defined in the database	IN
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The sessionReconnect() function returns the sessionKey of a session from which the user was disconnected previously without closing it

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):

UMS::reconnect

3.2.4 Service sessionClose

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN

Parameter	Type	Serialized type	Description	Mode
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The sessionClose() function closes the session identfied by the session key

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):

UMS::close

3.2.5 Service sessionList

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
listsession	string	ListSessions	listsession is the list of sessions	OUT
options	string	ListSessionOptions	allows the user to list sessions using several optional criteria such as: the state of sessions (active or inactive, by default, all sessions are listed), a period, a specific session or for admin to list all sessions of all users or sessions of a specific user.	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The sessionList() function lists all sessions of the user

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):

UMS::listSessions

3.2.6 Service userCreate

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
user	string	User	The user object	INOUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userCreate() function adds a new VISHNU user

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):

UMS::addUser

3.2.7 Service userUpdate

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
user	string	User	The user object	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userUpdate() function updates the user information except the userId and the password

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):

UMS::updateUser

3.2.8 Service userDelete

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
userId	string	n/a	userId represents the VISHNU user identifier of the user who will be deleted from VISHNU	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userDelete() function removes a user from VISHNU

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):

UMS::deleteUser

3.2.9 Service userList

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the identifier of the session generated by VISHNU	IN
userIdOption	string	n/a	allows an admin to get information about a specific user identified by his/her userId	IN
listuser	string	ListUsers	listuser is the list of users	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userList() function lists VISHNU 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):

UMS::listUsers

3.2.10 Service userPasswordChange

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
userId	string	n/a	userId represents the VISHNU user identifier	IN
password	string	n/a	password represents the password of the user	IN
passwordNew	string	n/a	passwordNew represents the new password of the user	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userPasswordChange() function changes the password

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):

UMS::changePassword

3.2.11 Service userPasswordReset

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
userId	string	n/a	userId represents the VISHNU user identifier of the user whose password will be reset	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The userPasswordReset() function resets the password of a user

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):

UMS::resetPassword

3.2.12 Service localAccountCreate

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
newAccount	string	LocalAccount	newAccount is the object which encapsulates the new local user configuration	IN
sshPublicKey	string	n/a	The SSH public key generated by VISHNU for accessing a local account	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The localAccountCreate() function adds a new local user configuration

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):

UMS::addLocalAccount

3.2.13 Service localAccountUpdate

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
LocalAccUpd	string	LocalAccount	is an object which encapsulates the local user configuration changes except the machineId and the userId	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The localAccountUpdate() function updates a local user configuration

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):

UMS::updateLocalAccount

3.2.14 Service localAccountDelete

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
userId	string	n/a	userId represents the VISHNU user identifier of the user whose local configuration will be deleted for the given machine	IN
machineId	string	n/a	machineId represents the identifier of the machine whose local configuration will be deleted for the given user	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The localAccountDelete() function removes a local user configuration (for a given user on a given machine) from VISHNU

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):

UMS::deleteLocalAccount

3.2.15 Service localAccountList

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
listLocalAcct	string	ListLocalAccounts	listLocalAccount is the list of the local user configuations	OUT
options	string	ListLocalAccOptions	allows an admin to list all local configurations of all users or a simple user to list his/her local user configurations on a specific machine	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The localAccountList() function lists the local user configurations

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):

UMS::listLocalAccounts

3.2.16 Service configurationSave

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
configuration	string	Configuration	The configuration is an object which encapsulates the configuration description	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The configurationSave() function saves the configuration of VISHNU

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):

UMS::saveConfiguration

3.2.17 Service configurationRestore

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
configuration	string	Configuration	The configuration is the object which encapsulates the configuration information	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The configurationRestore() function restores the configuration of VISHNU

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):

UMS::restoreConfiguration

3.2.18 Service machineCreate

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
machine	string	Machine	Machine information	INOUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The machineCreate() function adds a new machine in VISHNU

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):

UMS::addMachine

3.2.19 Service machineUpdate

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
machine	string	Machine	existing machine information	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The machineUpdate() function updates a machine description

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):

UMS::updateMachine

3.2.20 Service machineDelete

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
machineId	string	n/a	machineId represents the identifier of the machine	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The machineDelete() function removes a machine from VISHNU

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):

UMS::deleteMachine

3.2.21 Service machineList

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
options	string	ListMachineOptions	allows a user to list all VISHNU machines or information about a specific machine and an admin to list machines used by a specific user	IN
listMachine	string	ListMachines	listLocalAccount is the list of the local configs	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The machineList() function lists the machines in which the local user configurations are defined for the given user

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):

UMS::listMachines

3.2.22 Service optionValueList

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the identifier of the session generated by VISHNU	IN
options	string	ListOptOptions	allows to list a specific option or all default options values or for an admin to list options of a specific user	IN
listOptValues	string	ListOptionsValues	listOptValues is an object which encapsulates the list of options	OUT
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The optionValueList() function lists the options of the user

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):

UMS::listOptions

3.2.23 Service optionValueSet

Access

This service can be used by any VISHNU user

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
optionValue	string	OptionValue	The optionValue is an object which encapsulates the option information	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The optionValueSet() function configures an option of the user

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):

UMS::configureOption

3.2.24 Service optionValueSetDefault

Access

This service can be used by ADMIN users only

Parameters

Parameter	Type	Serialized type	Description	Mode
sessionKey	string	n/a	The sessionKey is the encrypted identifier of the session generated by VISHNU	IN
optionValue	string	OptionValue	The optionValue is an object which encapsulates the option information	IN
errorInfo	string	n/a	Additional information provided when an error code is returned	OUT

Description

The optionValueSetDefault() function configures a default option value

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):

UMS::configureDefaultOption

Chapter 4

Internal class and data structures

4.1 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 *UMS Client* component.
- **Server modelization**: describes the classes used to implement the *UMS Server* component.
- **Data modelization**: describes the data structure used to implement the *UMS Client* component and the *UMS Server* component.
- **Vishnu core functions modelization**: describes the classes and data structures used to implement the the VISHNU cross-modules components (components that are used by UMS and other VISHNU modules).

4.2 UMS client modelization

4.2.1 Class diagrams

4.2.1.1 UMS Client Class Diagram

This diagram describes all classes used to communicate with VISHNU System. Each class proxy contains the corresponding data class illustrated on the UMS Data modelization section and the methods usable by the UMS Client component. A QueryProxy class implements a generic model to list objects of VISHNU.

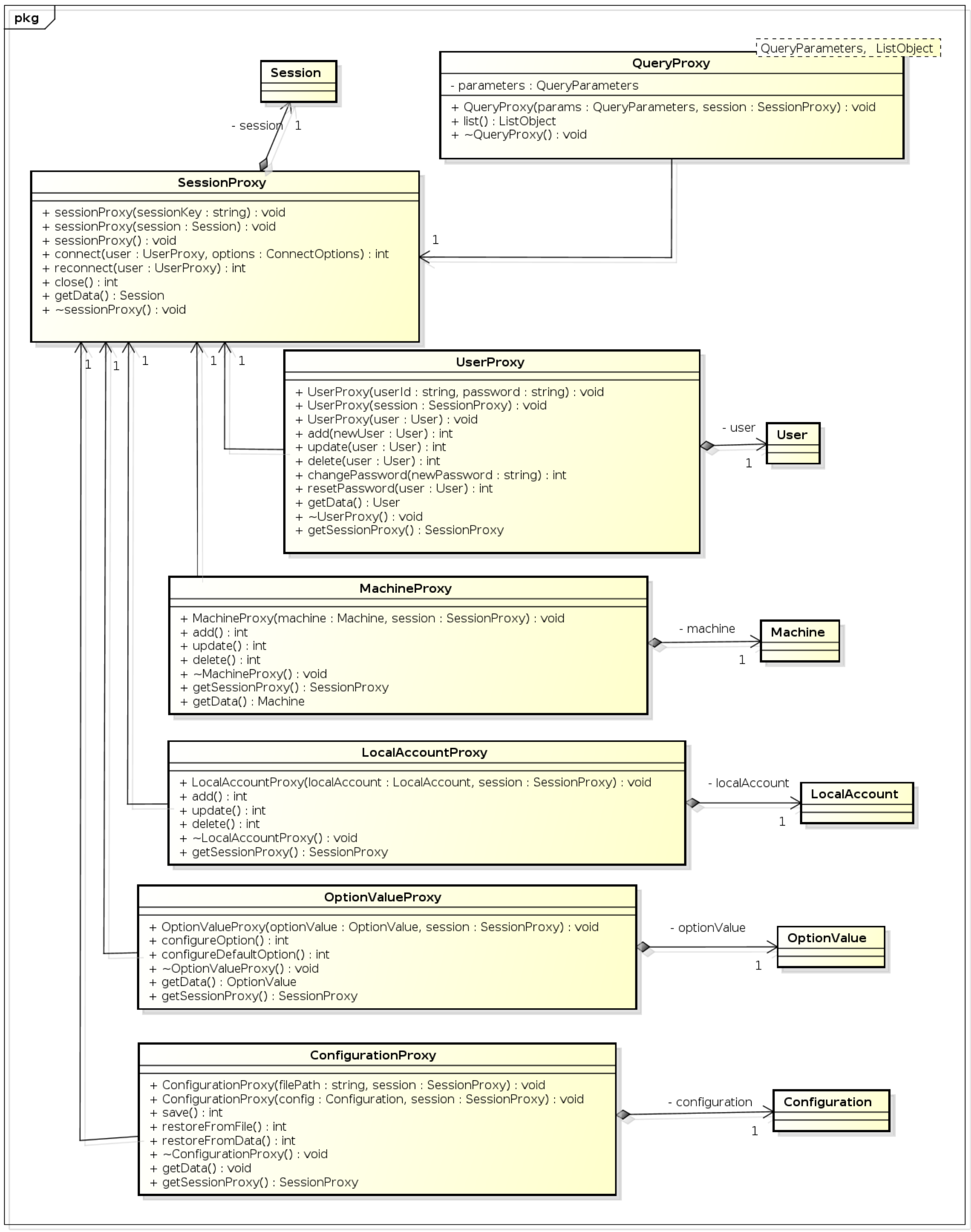
pkg
QueryParameters, ListObject
QueryProxy
- parameters : QueryParameters
+ QueryProxy(params : QueryParameters, session : SessionProxy) : void
+ list() : ListObject
+ ~QueryProxy() : void
Session
- session
1
SessionProxy
+ sessionProxy(sessionKey : string) : void
+ sessionProxy(session : Session) : void
+ sessionProxy() : void
+ connect(user : UserProxy, options : ConnectOptions) : int
+ reconnect(user : UserProxy) : int
+ close() : int
+ getData() : Session
+ ~sessionProxy() : void
1
UserProxy
+ UserProxy(userId : string, password : string) : void
+ UserProxy(session : SessionProxy) : void
+ UserProxy(user : User) : void
+ add(newUser : User) : int
+ update(user : User) : int
+ delete(user : User) : int
+ changePassword(newPassword : string) : int
+ resetPassword(user : User) : int
+ getData() : User
+ ~UserProxy() : void
+ getSessionProxy() : SessionProxy
- user
User
MachineProxy
+ MachineProxy(machine : Machine, session : SessionProxy) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~MachineProxy() : void
+ getSessionProxy() : SessionProxy
+ getData() : Machine
- machine
Machine
LocalAccountProxy
+ LocalAccountProxy(localAccount : LocalAccount, session : SessionProxy) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~LocalAccountProxy() : void
+ getSessionProxy() : SessionProxy
- localAccount
LocalAccount
OptionValueProxy
+ OptionValueProxy(optionValue : OptionValue, session : SessionProxy) : void
+ configureOption() : int
+ configureDefaultOption() : int
+ ~OptionValueProxy() : void
+ getData() : OptionValue
+ getSessionProxy() : SessionProxy
- optionValue
OptionValue
ConfigurationProxy
+ ConfigurationProxy(filePath : string, session : SessionProxy) : void
+ ConfigurationProxy(config : Configuration, session : SessionProxy) : void
+ save() : int
+ restoreFromFile() : int
+ restoreFromData() : int
+ ~ConfigurationProxy() : void
+ getData() : void
+ getSessionProxy() : SessionProxy
- configuration
Configuration

Figure 4.1: UMS Client Class Diagram

4.3 UMS server modelization

4.3.1 Class diagrams

4.3.1.1 UMS Server Class Diagram

This diagram presents the main objects used by UMS server component to process the UMS Client component requests. Each object that can be listed have a static method list with the corresponding options.

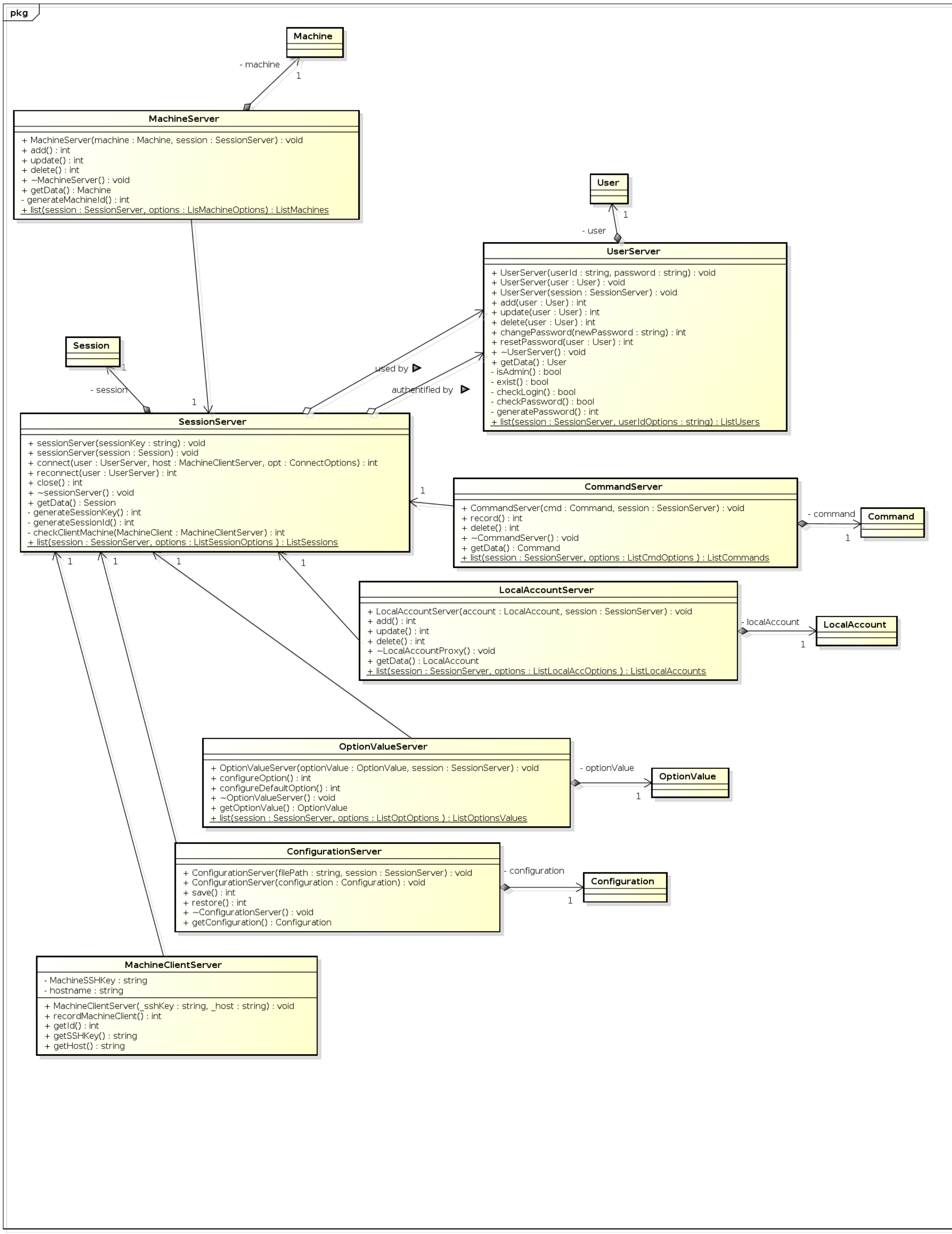
pkg
Machine
- machine
1
MachineServer
+ MachineServer(machine : Machine, session : SessionServer) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~MachineServer() : void
+ getData() : Machine
- generateMachineId() : int
+ list(session : SessionServer, options : LisMachineOptions) : ListMachines
User
1
- user
UserServer
+ UserServer(userId : string, password : string) : void
+ UserServer(user : User) : void
+ UserServer(session : SessionServer) : void
+ add(user : User) : int
+ update(user : User) : int
+ delete(user : User) : int
+ changePassword(newPassword : string) : int
+ resetPassword(user : User) : int
+ ~UserServer() : void
+ getData() : User
- isAdmin() : bool
- exist() : bool
- checkLogin() : bool
- checkPassword() : bool
- generatePassword() : int
+ list(session : SessionServer, userIdOptions : string) : ListUsers
Session
1
- session
1
used by
authentified by
SessionServer
+ sessionServer(sessionKey : string) : void
+ sessionServer(session : Session) : void
+ connect(user : UserServer, host : MachineClientServer, opt : ConnectOptions) : int
+ reconnect(user : UserServer) : int
+ close() : int
+ ~sessionServer() : void
+ getData() : Session
- generateSessionKey() : int
- generateSessionId() : int
- checkClientMachine(MachineClient : MachineClientServer) : int
+ list(session : SessionServer, options : ListSessionOptions) : ListSessions
1
CommandServer
+ CommandServer(cmd : Command, session : SessionServer) : void
+ record() : int
+ delete() : int
+ ~CommandServer() : void
+ getData() : Command
+ list(session : SessionServer, options : ListCmdOptions) : ListCommands
- command
Command
1
1
1
1
1
LocalAccountServer
+ LocalAccountServer(account : LocalAccount, session : SessionServer) : void
+ add() : int
+ update() : int
+ delete() : int
+ ~LocalAccountProxy() : void
+ getData() : LocalAccount
+ list(session : SessionServer, options : ListLocalAccOptions) : ListLocalAccounts
- localAccount
LocalAccount
1
OptionValueServer
+ OptionValueServer(optionValue : OptionValue, session : SessionServer) : void
+ configureOption() : int
+ configureDefaultOption() : int
+ ~OptionValueServer() : void
+ getOptionValue() : OptionValue
+ list(session : SessionServer, options : ListOptOptions) : ListOptionsValues
- optionValue
OptionValue
1
ConfigurationServer
+ ConfigurationServer(filePath : string, session : SessionServer) : void
+ ConfigurationServer(configuration : Configuration) : void
+ save() : int
+ restore() : int
+ ~ConfigurationServer() : void
+ getConfiguration() : Configuration
- configuration
Configuration
1
MachineClientServer
- MachineSSHKey : string
- hostname : string
+ MachineClientServer(_sshKey : string, _host : string) : void
+ recordMachineClient() : int
+ getId() : int
+ getSSHKey() : string
+ getHost() : string

Figure 4.2: UMS Server Class Diagram

4.4 UMS data modelization

4.4.1 Class diagrams

4.4.1.1 UMS Data Class Diagram

This diagram illustrates the structure and the relationship between data manipulated by the components Client and Server.

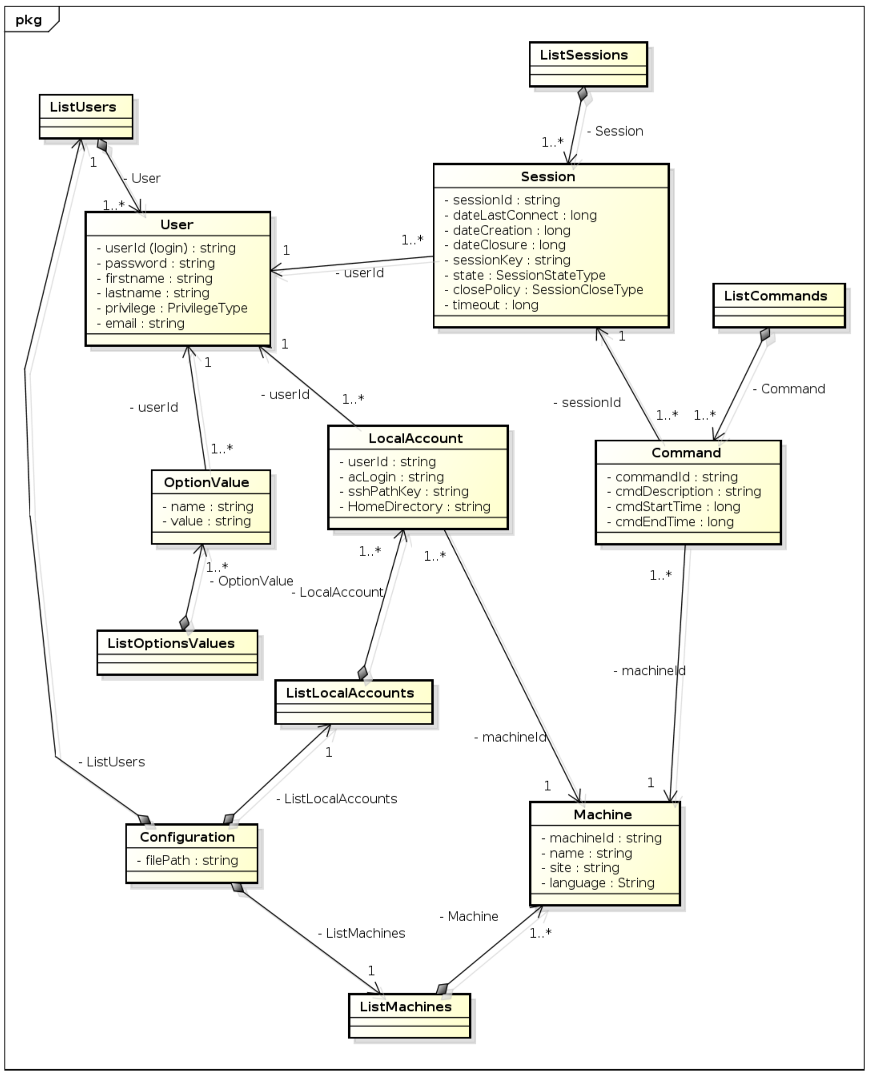

Figure 4.3: UMS Data Class Diagram

4.5 Vishnu core functions modelization

4.5.1 Introduction

The following elements describe the core classes (i.e. the classes that will be used by each module such as the exceptions and the databases). The modelization diagrams are given with some explanations about them.

4.5.2 Tables relationships

In order to have a coherent System, we have designed a relational model for the database. We need only one database that can contain all the Vishnu tables. The model is represented in figure 4.4. The rectangles are the tables and the lines represent the links between the tables.

The links between the tables are based on the following rules:

- – The *VISHNU* table has one or more *MACHINE*
  – A *MACHINE* is in one and only one *VISHNU* infrastructure
- – A *MACHINE* has one or more *CPU*
  – A *CPU* is in one and only one *MACHINE*
- – A *MACHINE* has one or more *DESCRIPTION*
  – A *DESCRIPTION* is for one and only one *MACHINE*
- – A *MACHINE* has one or more *THRESHOLD*
  – A *THRESHOLD* is for one and only one *MACHINE*
- – A *MACHINE* has one or more *ACCOUNT*
  – An *ACCOUNT* is for one and only one *MACHINE*
- – The *VISHNU* table has one or more *USER*
  – An *USER* is in one and only one *VISHNU* infrastructure
- – An *USER* has one or more *ACCOUNT*
  – An *ACCOUNT* is for one and only one *USER*
- – An *USER* has one or more *FILE TRANSFER*
  – A *FILE TRANSFER* is for one and only one *USER*
- – An *USER* has one or more *OPTION VALUE*
  – An *OPTION VALUE* is for one and only one *USER*
- – An *USER* sets one or more *THRESHOLD*
  – A *THRESHOLD* is set by one and only one *USER*
- – An *OPTION* has one or more *OPTION VALUE*
  – An *OPTION VALUE* is for one and only one *OPTION*
- – An *USER* has one or more *SESSION*
  – A *SESSION* is for one and only one *USER*
- – A *SESSION* has one or more *COMMAND*
  – A *COMMAND* is for one and only one *SESSION*
- – A *CLIENT MACHINE* has one or more *SESSION*
  – A *SESSION* is for one and only one *CLIENT MACHINE*
- – A *MACHINE* has one or more *STATE*
  – A *STATE* is for one and only one *MACHINE*
- – A *COMMAND* can have one or more *JOB*
  – A *JOB* is for one and only one *COMMAND*
- – A *COMMAND* can have one or more *FILE*
  – A *FILE* is for one and only one *COMMAND*

4.5.3 Relational model

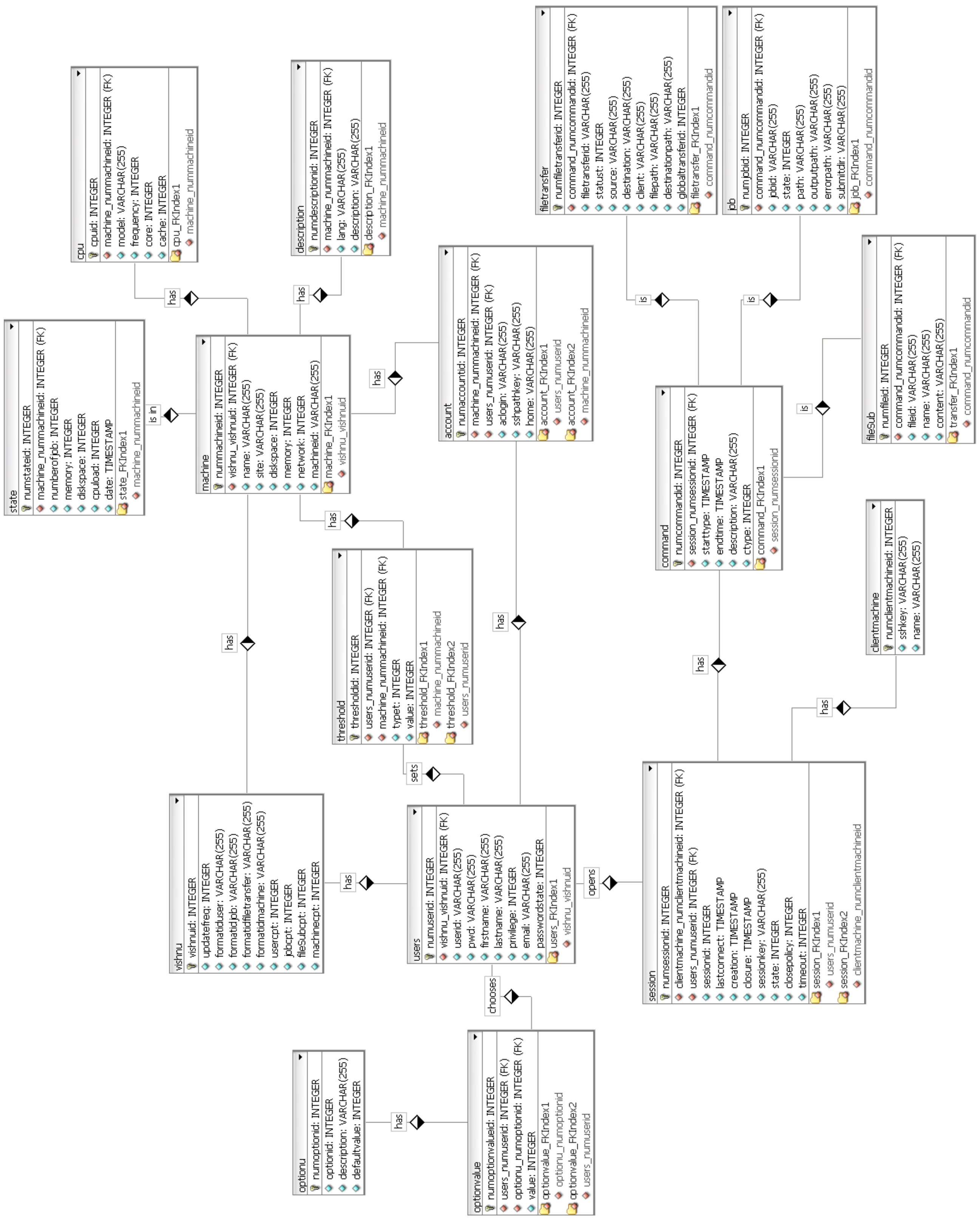

Figure 4.4: Relational model

4.5.4 The modelization

4.5.4.1 The database classes

The database class diagram is very simple. There is a database interface that defines a set of public operations that can be done over a database:

- commit
- rollback
- execute a query
- connect
- disconnect ...

And there are two examples of classes that implement the database. There is also a factory that can create the databases. See the diagram 4.5.

4.5.4.2 The exception classes

The exception class diagram defines a generic exception class, *VishnuException* that represents a generic exception that can be raised by a Vishnu function. This class has two subclasses, the *SystemException* that represents an exception due to a system problem and the *UserException* that represents an exception due to the user of the function (bad parameters typically). Both the server and clients have this way of building the exceptions. The *SystemException* has more specific subclasses depending on the modules that raises them. A key function is the append one, that allows to add a message to an existing vishnu exception. Thus, crossing the various level of the call can append information messages to specify the context of the exception. See the diagram 4.6.

4.5.5 Class diagrams

4.5.5.1 DB class diagram

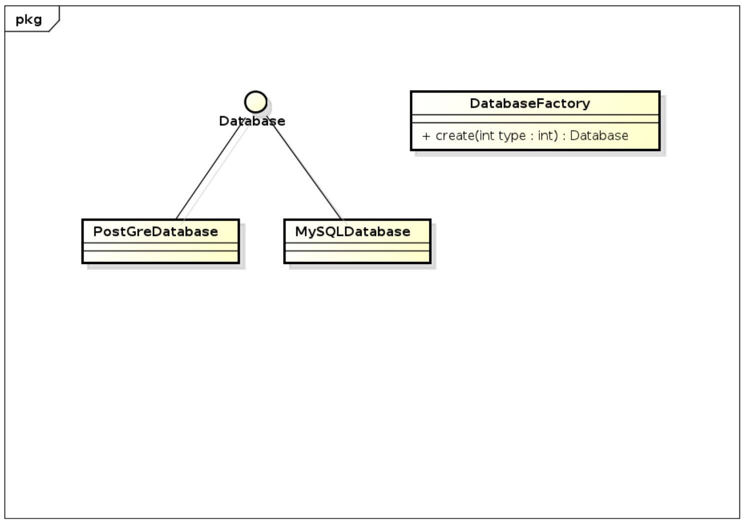

Figure 4.5: DB class diagram

4.5.5.2 exception

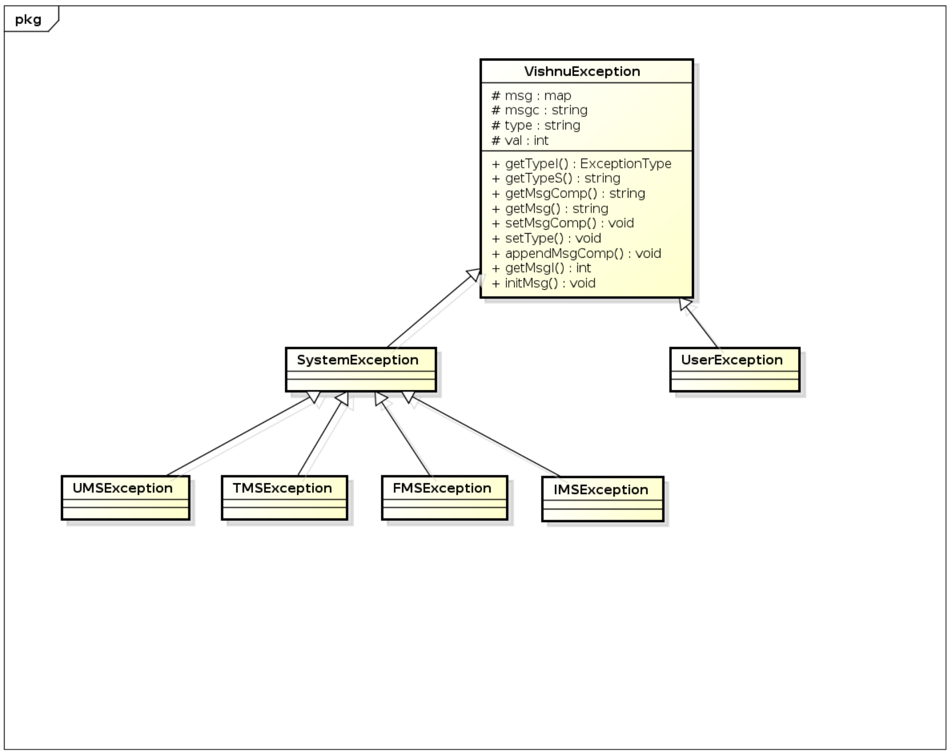

Figure 4.6: exception