

# **VISHNU D1.1 - APIs specifications**



**COLLABORATORS**

	<i>TITLE :</i> VISHNU D1.1 - APIs specifications		
<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	10 janvier 2011	

**REVISION HISTORY**

NUMBER	DATE	DESCRIPTION	NAME
0	05/01/2011	Formatting example	B.Isnard

# Table des matières

<b>1</b>	<b>Document presentation</b>	<b>1</b>
1.1	Document objectives . . . . .	1
1.2	Document structure . . . . .	1
1.3	Generic definition formats presentation . . . . .	1
1.3.1	Methods definition format . . . . .	1
1.3.1.1	Generic method definition format . . . . .	2
1.3.1.2	C++ specific aspects . . . . .	2
1.3.1.3	C++ specific aspects . . . . .	2
1.3.1.4	Web Services specific aspects . . . . .	2
1.3.2	Data types definition format . . . . .	3
1.3.2.1	Generic data definition format . . . . .	3
1.3.2.2	C++ specific aspects . . . . .	3
1.3.2.3	C++ specific aspects . . . . .	3
1.3.2.4	Web Services specific aspects . . . . .	3
1.4	References . . . . .	3
1.5	Glossary . . . . .	3
<b>2</b>	<b>API specification for User Management System (UMS)</b>	<b>4</b>
2.1	Definition of the functions of the package . . . . .	4
2.1.1	Function UMS : :connect . . . . .	4
2.1.2	Function UMS : :reconnect . . . . .	5
2.1.3	Function UMS : :addVishnuUser . . . . .	5
2.1.4	Function UMS : :updateUser . . . . .	6
2.1.5	Function UMS : :deleteUser . . . . .	7
2.1.6	Function UMS : :close . . . . .	7
2.1.7	Function UMS : :changePassword . . . . .	8
2.1.8	Function UMS : :resetPassword . . . . .	9
2.1.9	Function UMS : :addLocalAccount . . . . .	9
2.1.10	Function UMS : :updateLocalAccount . . . . .	10
2.1.11	Function UMS : :deleteLocalAccount . . . . .	10

2.1.12	Function UMS : :saveConfiguration . . . . .	11
2.1.13	Function UMS : :restoreConfiguration . . . . .	12
2.1.14	Function UMS : :addMachine . . . . .	12
2.1.15	Function UMS : :deleteMachine . . . . .	13
2.1.16	Function UMS : :listLocalAccount . . . . .	13
2.1.17	Function UMS : :listMachine . . . . .	14
2.1.18	Function UMS : :listOptions . . . . .	15
2.1.19	Function UMS : :listHistoryCmd . . . . .	15
2.1.20	Function UMS : :listUsers . . . . .	16
2.1.21	Function UMS : :listSessions . . . . .	17
2.1.22	Function UMS : :configureOption . . . . .	17
2.1.23	Function UMS : :configureDefaultOption . . . . .	18
2.1.24	Function UMS : :vishnuInitialize . . . . .	19
2.1.25	Function UMS : :vishnuFinalize . . . . .	19
2.2	Data types definitions . . . . .	20
<b>3</b>	<b>API specification for Tasks Management System (TMS)</b>	<b>25</b>
3.1	Definition of the functions of the package . . . . .	25
3.1.1	Function TMS : :submitJob . . . . .	25
3.1.2	Function TMS : :listJobs . . . . .	26
3.1.3	Function TMS : :getJobInfo . . . . .	27
3.1.4	Function TMS : :cancelJob . . . . .	27
3.1.5	Function TMS : :getJobOutPut . . . . .	28
3.1.6	Function TMS : :getAllJobsOutPut . . . . .	29
3.1.7	Function TMS : :listQueues . . . . .	30
3.1.8	Function TMS : :setMachineEnv . . . . .	30
3.1.9	Function TMS : :setMachineRefreshPeriod . . . . .	31
3.1.10	Function TMS : :getJobProgress . . . . .	31
3.2	Data types definitions . . . . .	32
<b>4</b>	<b>API specification for Information Management System (IMS)</b>	<b>36</b>
4.1	Definition of the functions of the package . . . . .	36
4.1.1	Function IMS : :getUpdateFrequency . . . . .	36
4.1.2	Function IMS : :export . . . . .	36
4.1.3	Function IMS : :replay . . . . .	37
4.1.4	Function IMS : :getMetricVal . . . . .	37
4.1.5	Function IMS : :getCurrentData . . . . .	38
4.1.6	Function IMS : :getProcesses . . . . .	39
4.1.7	Function IMS : :setSystemThreshold . . . . .	39

4.1.8	Function IMS : :getSystemThreshold . . . . .	40
4.1.9	Function IMS : :defineUserIdentifier . . . . .	40
4.1.10	Function IMS : :defineMachineIdentifier . . . . .	41
4.1.11	Function IMS : :defineJobIdentifier . . . . .	41
4.1.12	Function IMS : :defineTransferIdentifier . . . . .	42
4.1.13	Function IMS : :loadShed . . . . .	42
4.1.14	Function IMS : :setUpdateFrequency . . . . .	43
4.1.15	Function IMS : :restart . . . . .	43
4.1.16	Function IMS : :updateMachine . . . . .	44
4.2	Data types definitions . . . . .	45
<b>5</b>	<b>API specification for File Management System (FMS)</b>	<b>47</b>
5.1	Definition of the functions of the package . . . . .	47
5.1.1	Function FMS : :createFile . . . . .	47
5.1.2	Function FMS : :createDir . . . . .	48
5.1.3	Function FMS : :removeFile . . . . .	48
5.1.4	Function FMS : :removeDir . . . . .	49
5.1.5	Function FMS : :chGrpOfFile . . . . .	50
5.1.6	Function FMS : :chModOfFile . . . . .	50
5.1.7	Function FMS : :headOfFile . . . . .	51
5.1.8	Function FMS : :tailOfFile . . . . .	52
5.1.9	Function FMS : :listDir . . . . .	52
5.1.10	Function FMS : :copyFile . . . . .	53
5.1.11	Function FMS : :copyAsyncFile . . . . .	54
5.1.12	Function FMS : :moveFile . . . . .	55
5.1.13	Function FMS : :moveAsyncFile . . . . .	55
5.1.14	Function FMS : :stopFileTransfer . . . . .	56
5.1.15	Function FMS : :stopAllFileTransfer . . . . .	57
5.1.16	Function FMS : :stopAllUsersFileTransfer . . . . .	57
5.1.17	Function FMS : :getFileTransferStatus . . . . .	58
5.1.18	Function FMS : :listFilesTransferStatus . . . . .	58
5.1.19	Function FMS : :listAllUsersFilesTransferStatus . . . . .	59
5.1.20	Function FMS : :DisplayFilesTransferHistory . . . . .	60
5.1.21	Function FMS : :DisplayAllUsersFilesTransferHistory . . . . .	60
5.1.22	Function FMS : :GetFilesInfo . . . . .	61
5.2	Data types definitions . . . . .	62

# Chapitre 1

## Document presentation

### 1.1 Document objectives

This document presents the detailed specifications of the Vishnu APIs (Application Programming Interfaces). The following APIs are included in the project :

- C++ API
- Python (v2.x) API
- Web services (WSDL 1.1) API

These specifications include the definition of all methods and all data types in a format that is common to all APIs. Therefore the description is not tied to a particular implementation and all implementations will follow the same logic and will differ only when the language that is used imposes some constraints.

Specific aspects of each implementation language are described in the section 1.3.

### 1.2 Document structure

The document is divided into 4 parts corresponding to the four modules that compose the Vishnu system :

- UMS : Users Management System
- TMS : Tasks Management System
- FMS : Files Management System
- IMS : Information Management System

Each module corresponds to a chapter in the document, and each chapter contains the following sections :

- A first section describing the definition of all the methods provided by the library
- A second section describing the definition of all the data types provided by the library

### 1.3 Generic definition formats presentation

This section presents the formats used in the following chapters to describe the methods and data types provided by the libraries. It also details the particular implementation constraints for each language.

#### 1.3.1 Methods definition format

The following paragraphs show how all methods (or "operations" in the Web Services terminology) are specified in this document. First, the generic format used for each Vishnu module is explained, then the aspects that are specific to each implementation language are detailed.

### 1.3.1.1 Generic method definition format

#### Parameters

The following table contain all the input and output parameters of the method, along with their type and description, and their optional or required flag.

Parameter	Type	Description	Mode	Required
sessionKey	string	This is an example of a required input parameter	IN	yes
listOfJobs	ListJobs	This is an example of an output parameter that is always provided	OUT	yes

#### Access

Here is detailed the access level of the method 'myMethod' (i.e. the privilege required to use this method)

#### Description

Here is detailed the purpose of the method 'myMethod'

#### Return Value

Here are detailed the different return codes provided by the method. Please note that these return codes may be implemented differently depending on the language, for example by using an exception mechanism. In all implementations the library will provide a way of mapping the code to a human-readable message that will contain detailed information about the context of the exception that happened.

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

#### Signature

This shows the C++ signature of the method.

```
int myMethod(const string& sessionKey, ListJobs& listOfJobs);
```

### 1.3.1.2 C++ specific aspects

- The output parameters will be implemented as references in the method signature.
- The methods will always return an integer with a default value for success.
- The methods will throw exceptions for each error message specified. The exception will contain additional details provided by the server.

### 1.3.1.3 C++ specific aspects

- The output parameters will be implemented as a Python tuple returned by the method.

### 1.3.1.4 Web Services specific aspects

- The input and output parameters will be implemented as Java Beans : a "Request" bean containing the input parameters and a "Response" bean containing the output parameters.
- The methods will throw exceptions for each error message specified. The exception will contain additional details provided by the server.
- Methods with restricted access (administration) will not be included in the WS API.

### 1.3.2 Data types definition format

The following paragraphs show how all data types are specified in this document. First, the generic format used for each Vishnu data type is explained, then the aspects that are specific to each implementation language are detailed.

#### 1.3.2.1 Generic data definition format

**Class Module : :Class Content**

Name	Type	Description
Class attribute name	Class attribute type	Description/usage of the attribute

#### 1.3.2.2 C++ specific aspects

- All attributes of the class will be private.
- For each attribute of the class a couple of getter/setter methods will be implemented.
- The string type will be mapped to the C++ STL string type.
- 

#### 1.3.2.3 C++ specific aspects

- For each attribute of the class a couple of getter/setter methods will be implemented.
- The string type will be mapped to standard Python strings.
- 

#### 1.3.2.4 Web Services specific aspects

- When a single instance of object is used as input or output parameter, the attributes of the object will be mapped respectively to attributes of the 'Request' or 'Response' Java Bean.
- When multiple instances of object are used as input or output parameter (for example list of machines or list of users) the 'Request' or 'Response' Java Bean will contain a 'data' subclass containing the instances. This follows the standard WSDL/Java mapping for Apache-CXF.
- 

## 1.4 References

- **UMS.wsdl** : WSDL file for the UMS module
- **TMS.wsdl** : WSDL file for the TMS module
- **FMS.wsdl** : WSDL file for the FMS module
- **IMS.wsdl** : WSDL file for the IMS module

## 1.5 Glossary



## Chapitre 2

# API specification for User Management System (UMS)

### 2.1 Definition of the functions of the package

#### 2.1.1 Function UMS : :connect

##### Access

This function can be used by any Vishnu user.

##### Parameters

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier	IN	yes
password	string	password represents the password of the user	IN	yes
options	<b>ConnectOptions</b>	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 he/she was a specific user	IN	no
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	OUT	yes

##### Description

The connect() function opens a session

##### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NOT_AUTHENTICATED	Unknown user
UNKNOWN_CLOSURE_MODE	The name of the closure mode is unknown
INCORRECT_TIMEOUT	The value of the timeout is incorrect (negative or higher than the TIMEOUT treshold)
UNKNOWN_USERID	The userId is unknown
NO_ADMIN	The user is not an administrator
INCORRECT_PASSSSWORD_SIZE	The size of the password is incorrect

Name	Description
INCORRECT_USERID_SIZE	The size of the login is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

int **connect**(const string& userId, const string& password, const ConnectOptions& options=ConnectOptions(), string& session-Key);

### 2.1.2 Function UMS : :reconnect

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier	IN	yes
password	string	password represents the password of the user	IN	yes
sessionId	string	sessionId is the identifier of the session defined in the database	IN	yes
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	OUT	yes

### Description

The reconnect() function returns the sessionKey of a session in 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 function.

Name	Description
VISHNU_OK	The service was performed successfully
NOT_AUTHENTICATED	Unknown user
INCORRECT_USERID_SIZE	The size of the login is incorrect
INCORRECT_PASSSSWORD_SIZE	The size of the password is incorrect
SESSION_INCOMPATIBILITY	This session identifier is incompatible with the authenticated user
UNKNOWN_SESSION_ID	The session Id is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

int **reconnect**(const string& userId , const string& password, const string& sessionId, string& sessionKey);

### 2.1.3 Function UMS : :addVishnuUser

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
newUser	User	newUser is an object which encapsulates the new user information	IN	yes

### Description

The addVishnuUser() function adds a new VISHNU user

### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
INCORRECT_USERID_SIZE	The size of the login is incorrect
INCORRECT_PASSSSWORD_SIZE	The size of the password is incorrect
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
USERID_EXISTING	The userId already exists in the database
INVALID_MAIL_ADRESS	The mail address is invalId
USERID_REQUIRED	The userId must to be defined
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

```
int addVishnuUser(const string& sessionKey, const User& newUser);
```

## 2.1.4 Function UMS : :updateUser

### Access

This function can be used by ADMIN users only

### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	UpdateUserOptions	options is an object which encapsulates user information updated	IN	no
userId	string	represents the VISHNU user identifier of the user whose information will be updated	IN	yes

### Description

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

### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator

Name	Description
UNKNOWN_USERID	The userId is unknown
INVALID_MAIL_ADRESS	The mail address is invalid
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
INCORRECT_USERID_SIZE	The size of the login is incorrect
INCORRECT_PASSSSWORD_SIZE	The size of the password is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int updateUser(const string& sessionKey, const UpdateUserOptions& options=UpdateUserOptions(), const string& userId);
```

**2.1.5 Function UMS : :deleteUser****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier of the user who will be deleted from VISHNU	IN	yes
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes

**Description**

The deleteUser() function removes a user from VISHNU

**Return Value**

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

Name	Description
VISHNU_OK	The service was performed successfully
UNKNOWN_USERID	The userId is unknown
NO_ADMIN	The user is not an administrator
INCORRECT_USERID_SIZE	The size of the login is incorrect
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int deleteUser(const string& userId, const string& sessionKey);
```

**2.1.6 Function UMS : :close****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes

### Description

The close() function closes the session identified by the session key

### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
COMMAND_RUNNING	Command(s) is/are running
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database

### Signature

```
int close(const string& sessionKey);
```

## 2.1.7 Function UMS : :changePassword

### Access

This function can be used by any Vishnu user.

### Parameters

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier	IN	yes
password	string	password represents the password of the user	IN	yes
passwordNew	string	passwordNew represents the new password of the user	IN	yes

### Description

The changePassword() function changes the password

### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NOT_AUTHENTICATED	Unknown user
INCORRECT_USERID_SIZE	The size of the login is incorrect
INCORRECT_PASSSSWORD_SIZE	The size of the password is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

```
int changePassword(const string& userId, const string& password, const string& passwordNew);
```

### 2.1.8 Function UMS : :resetPassword

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier of the user whose password will be reset	IN	yes
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes

#### Description

The resetPassword() function resets the password of a user

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
UNKNOWN_USERID	The userId is unknown
INCORRECT_USERID_SIZE	The size of the login is incorrect
INCORRECT_PASSWORD_SIZE	The size of the password is incorrect
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int resetPassword(const string& userId, const string& sessionKey);
```

### 2.1.9 Function UMS : :addLocalAccount

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
newAccount	LocalAccount	newAccount is the object which encapsulates the new local user configuration	IN	yes

#### Description

The addLocalAccount() function adds a new local user configuration

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
LOCAL_ACCOUNT_EXIST	The local account already exists for the given user on the given machine
USERID_REQUIRED	The userId must to be defined
MACHINEID_REQUIRED	The machineId must to be defined
UNKNOWN_USERID	The userId is unknown
UNKNOWN_MACHINE	The machineId is unknown
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int addLocalAccount(const string& sessionKey, const LocalAccount& newAccount);
```

**2.1.10 Function UMS : :updateLocalAccount****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
LocalAccUpd	LocalAccount	is an object which encapsulates the local user configuration changes except the machineId and the userId	IN	yes

**Description**

The updateLocalAccount() function updates a local user configuration

**Return Value**

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

Name	Description
UNKNOWN_USERID	The userId is unknown
UNKNOWN_MACHINE	The machineId is unknown
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
UNKNOWN_LOCAL_ACCOUNT	The local configuration for the given user on the given machine is unknown
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int updateLocalAccount(const string& sessionKey, const LocalAccount& LocalAccUpd);
```

**2.1.11 Function UMS : :deleteLocalAccount****Access**



This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
userId	string	userId represents the VISHNU user identifier of the user whose local configuration will be deleted for the given machine	IN	yes
machineId	string	machineId represents the identifier of the machine whose local configuration will be deleted for the given user	IN	yes
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes

#### Description

The deleteLocalAccount() 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 function.

Name	Description
VISHNU_OK	The service was performed successfully
UNKNOWN_LOCAL_ACCOUNT	The local configuration for the given user on the given machine is unknown
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int deleteLocalAccount(const string& userId, const string& machineId, const string& sessionKey);
```

### 2.1.12 Function UMS : :saveConfiguration

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
configuration	Configuration	The configuration is an object which encapsulates the configuration description	OUT	yes

#### Description

The saveConfiguration() function saves the configuration of VISHNU

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator



Name	Description
SAVE_CONFIG_ERROR	A problem occurs during the configuration saving
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

int saveConfiguration(const string& sessionKey, Configuration& configuration);

### 2.1.13 Function UMS : :restoreConfiguration

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
configuration	Configuration	The configuration is the object which encapsulates the configuration description	IN	yes

#### Description

The restoreConfiguration() function restores the configuration of VISHNU

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
RESTORE_CONFIG_ERROR	A problem occurs during the configuration restoring
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

int restoreConfiguration(const string& sessionKey, const Configuration& configuration);

### 2.1.14 Function UMS : :addMachine

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
newMachine	Machine	newMachine is an object which encapsulates the new machine information	IN	yes

#### Description

The addMachine() function adds a new machine in VISHNU

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
MACHINE_EXISTING	The machineId already exists in the database
MACHINEID_REQUIRED	The machineId must to be defined
NO_ADMIN	The user is not an administrator
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int addMachine(const string& sessionKey, const Machine& newMachine);
```

### 2.1.15 Function UMS : :deleteMachine

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	machineId represents the identifier of the machine	IN	yes
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes

#### Description

The deleteMachine() function removes a machine from VISHNU

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
UNKNOWN_MACHINE	The machineId is unknown
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int deleteMachine(const string& machineId, const string& sessionKey);
```

### 2.1.16 Function UMS : :listLocalAccount

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	ListLocalAccOptions	allows an admin to list all local configurations of all users in the database or a user to list a local user configuration on a specific machine	IN	no
listLocalAcct	ListLocalAccounts	listLocalAccount is the list of the local user configurations	OUT	yes

#### Description

The listLocalAccount() function lists the local user configurations

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
UNKNOWN_USERID	The userId is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int listLocalAccount(const string& sessionKey, const ListLocalAccOptions& options=ListLocalAccOptions(), ListLocalAccounts& listLocalAcct);
```

### 2.1.17 Function UMS : :listMachine

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	ListMachineOptions	allows an admin to list all machines in the database or a user to list information about a specific machine	IN	no
listMachine	ListMachines	listLocalAccount is the list of the local configs	OUT	yes

#### Description

The listMachine() function lists the machines in which 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 function.

Name	Description
VISHNU_OK	The service was performed successfully

Name	Description
UNKNOWN_USERID	The userId is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

int **listMachine**(const string& sessionKey, const ListMachineOptions& options=ListMachineOptions(), ListMachines& listMachine);

## 2.1.18 Function UMS : :listOptions

### Access

This function can be used by any Vishnu user.

### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	ListOptOptions	allows the user to list a specific option or an admin to list all options of the System or to list all options of a specific user	IN	no
listOptValues	ListOptionsValues	listOptions is the list of options	OUT	yes

### Description

The listOptions() function lists the options of the user

### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
UNKNOWN_USERID	The userId is unknown
UNKNOWN_OPTION	the name of the option is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

### Signature

int **listOptions**(const string& sessionKey, const ListOptOptions& options=ListOptOptions(), ListOptionsValues& listOptValues=ListOptionsValues());

## 2.1.19 Function UMS : :listHistoryCmd

### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	ListCmdOptions	allows the user to list commands using several optional criteria : a period, specific session and for admin to list all commands of all users in the database or commands from a specific user	IN	no
listCommands	ListCommands	listCommands is the list of commands	OUT	yes

#### Description

The listHistoryCmd() function lists the commands

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
UNKNOWN_USERID	The userId is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int listHistoryCmd(const string& sessionKey, const ListCmdOptions& options=ListCmdOptions(), ListCommands& listCommands);
```

### 2.1.20 Function UMS : :listUsers

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
userIdOption	string	allows an admin to list one user identified by his/her userId	IN	no
listuser	ListUsers	listuser is the list of users	OUT	yes

#### Description

The listUsers() function lists users

#### Return Value

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

Name	Description
NO_ADMIN	The user is not an administrator

Name	Description
UNKNOWN_USERID	The userId is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int listUsers(const string& sessionKey, const string& userIdOption, ListUsers& listuser);
```

**2.1.21 Function UMS : :listSessions****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
options	ListSessionOptions	allows the user to list sessions using several optional criteria such as : the state of sessions (actives or inactives, 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	no
listsession	ListSessions	listsession is the list of sessions	OUT	yes

**Description**

The listSessions() function lists all sessions of the user

**Return Value**

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
UNKNOWN_USERID	The userId is unknown
UNKNOWN_SESSION_OPTION	The name of the session option is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
INCORRECT_DATE_OPTION	The date option is incorrect
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

**Signature**

```
int listSessions(const string& sessionKey, const ListSessionOptions& options=ListSessionOptions(), ListSessions& listsession);
```

**2.1.22 Function UMS : :configureOption****Access**

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
optionValue	OptionValue	The optionValue is an object which encapsulates the option information	IN	yes

#### Description

The configureOption() function configures an option

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
UNKNOWN_OPTION	the name of the option is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available

#### Signature

```
int configureOption(const string& sessionKey, const OptionValue& optionValue);
```

### 2.1.23 Function UMS : :configureDefaultOption

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
optionValue	OptionValue	The optionValue is an object which encapsulates the option information	IN	yes

#### Description

The configureDefaultOption() function configures a default option value

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully
NO_ADMIN	The user is not an administrator
UNKNOWN_OPTION	the name of the option is unknown
SESSIONKEY_EXPIRED	The sessionKey is expired. The session is closed.
SESSIONKEY_NOT_FOUND	The sessionKey is unrecognized
DB_ERROR	A problem occurs with the database
UMS_SERVER_NOT_AVAILABLE	The server UMS is not available



**Signature**

```
int configureDefaultOption(const string& sessionKey, const OptionValue& optionValue);
```

**2.1.24 Function UMS : :vishnuInitialize****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
configPath	string	configPath is the path of VISHNU configuration file	IN	yes

**Description**

The vishnuInitialize() function initializes VISHNU

**Return Value**

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

Name	Description
VISHNU_OK	The service was performed successfully
INCORRECT_CFG_PATH	The configuration file path is incorrect

**Signature**

```
int vishnuInitialize(const string& configPath);
```

**2.1.25 Function UMS : :vishnuFinalize****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
-----------	------	-------------	------	----------

**Description**

The vishnuFinalize() function allows a user to go out properly from VISHNU

**Return Value**

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

Name	Description
VISHNU_OK	The service was performed successfully
VISHNU_FINALIZE_ERROR	An error occurs during vishnuFinalize

**Signature**

```
int vishnuFinalize();
```



## 2.2 Data types definitions

### ConnectOptions Class Content

Name	Type	Description
closePolicy	SessionCloseType	is an option for closing session automatically
sessionInactivityDelay	int	The sessionInactivityDelay is the maximum delay between two API commands when the CLOSE_ON_TIMEOUT option is set
substituteUserId	string	is an option which allows an admin to open a session as he/she was a specific user identified by his/her userId

### SessionCloseType Enumeration Type

Name	Value
CLOSE_ON_DISCONNECT	0
CLOSE_ON_TIMEOUT	1

### User Class Content

Name	Type	Description
userId	string	represents the VISHNU user identifier
password	string	is the password of the user. At the beginning, an admin can give a temporary password or it is automatically generated by the System.
firstname	string	is the firstname of the user
lastname	string	is the lastname of the user
privilege	int	is the privilege of the user (admin or simple user)
email	string	is the email of the user

### UpdateUserOptions Class Content

Name	Type	Description
firstname	string	represents the updated firstname of the user
lastname	string	represents the updated lastname of the user
privilege	string	represents the updated privilege of the user
email	string	represents the updated email adress of the user

### LocalAccount Class Content

Name	Type	Description
userId	string	The userId represents the VISHNU user identifier of the user of the local user configuration
machineId	string	The MachineId represents the identifier of the machine associated to the local user configuration
acLogin	string	accLogin represents the login of the user on the associated machine
sshKeyPath	string	sshKeyPath is the path of the ssh key of the user on the associated machine
HomeDirectory	string	HomeDirectory is the path of the home directory of the user on the associated machine

### Configuration Class Content

Name	Type	Description
descConf	string	represents the description of the configuration
listConfUsers	List of <b>User</b>	is the list of users objects
listConfMachines	List of <b>Machine</b>	is a list of machines objects
listConfLocalAccounts	List of <b>LocalAccount</b>	is the list of LocalAccount objects

#### Machine Class Content

Name	Type	Description
machineId	string	represents the identifier of the machine
name	string	represents the name of the machine
site	string	represents the location of the machine
totalDiskSpace	string	represents the total disk space of the machine
totalMemory	string	represents the total memory of the machine
machineDescription	string	represents the description of the machine
language	string	represents the language in which the description has been done

#### ListLocalAccOptions Class Content

Name	Type	Description
AdminListOption	boolean	is an admin option for listing all local configurations of all users
userId	string	is an admin option for listing the local configurations of a specific user
machineId	string	is an option for listing local user configurations on a specific machine

#### ListLocalAccounts Class Content

Name	Type	Description
accounts	List of <b>LocalAccount</b>	is a list of LocalAccount objects which encapsulates local user configurations

#### ListMachineOptions Class Content

Name	Type	Description
userId	string	is an admin option for listing machines in which a specific user has a local configuration
ListAllOption	boolean	is an admin option for listing all machine defined in the database
machineId	string	is an option for listing information about a specific machine

#### ListMachines Class Content

Name	Type	Description
machines	List of <b>Machine</b>	is a list of machines objects which encapsulates the machines information

#### ListOptOptions Class Content

Name	Type	Description
AdminListOption	boolean	is an admin option for listing all user options defined in VISHNU
userId	string	is an admin option for listing the options of a specific user
optionName	string	allows the user to list a specific option identified by its name

#### ListOptionsValues Class Content

Name	Type	Description
optionValues	List of <b>OptionValue</b>	is a list of optionValue objects which encapsulates the optionValue information

#### OptionValue Class Content

Name	Type	Description
optionName	string	represents the name of an option
value	string	represents the value of an option

#### ListCmdOptions Class Content

Name	Type	Description
AdminListOption	boolean	is an admin option for listing all commands of all users
userId	string	is an admin option for listing commands launched by a specific user identified by his/her userId
sessionId	string	lists all commands launched within a specific session
startDateOption	long	allows the user to organize the commands listed by providing the start date (the UNIX timestamp of the start date is used)
endDateOption	long	allows the user to organize the commands listed by providing the end date (the timestamp of the end date is used). By default, the end date is the current day

#### ListCommands Class Content

Name	Type	Description
Commands	List of <b>Command</b>	is the list of commands objects

#### Command Class Content

Name	Type	Description
commandId	string	is the identifier of a command
sessionId	string	The sessionId is the identifier of the session define in the database
machineId	string	The machineId is the identifier of the machine used by the command
cmdDescription	string	cmdDescription is the description of the command
cmdStartTime	long	cmdStartTime is the date of the command beginning (the UNIX timestamps is used)
cmdEndTime	long	cmdEndTime is the date of the command end (the UNIX timestamps is used)

**ListUsers Class Content**

Name	Type	Description
users	List of <b>User</b>	is the list of users objects

**ListSessionOptions Class Content**

Name	Type	Description
sessionListOption	<b>SessionStateType</b>	specifies the type of the sessions which will be listed (INACTIVE or ACTIVE)
sessionClosePolicy	<b>SessionCloseType</b>	specifies the closure mode of the sessions which will be listed (CLOSE_ON_TIMEOUT or CLOSE_ON_DISCONNECT)
sessionInactivityDelay	int	specifies the inactivity delay of the sessions which will be listed
machineId	string	allows the user to list sessions opened on a specific machine
AdminListOption	boolean	is an admin option for listing all sessions of all users
userId	string	is an admin option for listing sessions opened by a specific user
sessionId	string	allows the user to list all commands launched within a specific session
startDateOption	long	allows the user to organize the commands listed by providing the start date (the UNIX timestamp of the start date is used)
endDateOption	long	allows the user to organize the commands listed by providing the end date (the timestamp of the end date is used). By default, the end date is the current day

**SessionStateType Enumeration Type**

Name	Value
INACTIVE	0
ACTIVE	1

**ListSessions Class Content**

Name	Type	Description
sessions	List of <b>Session</b>	is the list of session objects

**Session Class Content**

Name	Type	Description
sessionId	string	represents the VISHNU session identifier of the session
userId	string	represents the VISHNU user identifier of the user who has opened the session
sessionKey	string	is the key of the session generated by VISHNU
dateLastConnect	long	is the date of the last connection to the session (the UNIX timestamps is used)
dateCreation	long	is the date of the first connection to the session (the UNIX timestamps is used)
dateClosure	long	is the date of the closure of the session (the UNIX timestamps is used)
state	<b>SessionStateType</b>	is the state of the session (ACTIVE OR INACTIVE)

Name	Type	Description
closePolicy	<b>SessionCloseType</b>	is the way to close the session
timeout	long	is the inactivity delay in seconds associated to the CLOSE_ON_TIMEOUT option

## Chapitre 3

# API specification for Tasks Management System (TMS)

### 3.1 Definition of the functions of the package

#### 3.1.1 Function TMS : :submitJob

##### Access

This function can be used by any Vishnu user.

##### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Is the id of the machine where the job must be submitted	IN	yes
scriptFilePath	string	The path to the file containing the characteristics (job command, and batch scheduler directive required or optional) of the job to submit.	IN	yes
jobId	string	Is the returned id of the submitted job	OUT	yes
jobPath	string	Is the path to the file containing job characteristics	OUT	yes
options	SubmitOptions	Is an instance of the class SubmitOptions. Each optionnal value is associated to a set operation (e.g : setNbCpu(...)) in the class SubmitOptions. If no set operation is not called on the instance object options, the job is submitted with the options defined in the scriptFilePath. Otherwise the job is submitted with the optionnal values set by the options object and optionnal values defined in the scriptFilePath, but optionnal values set by SubmitOptions object take precedence over those in scriptFilePath. With in the object options or within the scriptFilePath, the last occurrence of an optionnal value takes precedence over earlier occurrence.	IN	no

##### Description

The submitJob() function submits job on a machine through the use of a script (scriptFilePath). The script is a shell script which will be executed by a command shell such as sh or csh.

##### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully.
TMS_UNKNOWN_MACHINE	The machine is not known.
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_PATH	The path to the file containing the characteristics of the job to submit is not a valid path
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_SUBMIT_SERVICE_NOT_AVAILABLE	Indicates that the service to perform the submit operation is not found.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_QUEUE	Indicates that the specified queue by the user is not known.

### Signature

```
int submitJob(const string& sessionKey, const string& machineId, const string& scriptFilePath, string& jobId, string& jobPath,
const SubmitOptions& options=SubmitOptions());
```

### 3.1.2 Function TMS : :listJobs

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
listOfJobs	ListJobs	The constructed object list of jobs	OUT	yes
options	ListJobsOptions	Additional options for jobs listing	IN	no

#### Description

The listJobs() function gets a list of all submitted jobs

#### Return Value

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
VISHNU_OK	The service was performed successfully.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.



Name	Description
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

**Signature**

```
int listJobs(const string& sessionKey, const string& machineId, ListJobs& listOfJobs, const ListJobsOptions& options=ListJobsOptions)
```

**3.1.3 Function TMS : :getJobInfo****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
jobId	string	The id of the job	IN	yes
jobInfos	Job	The resulting information on the job	OUT	yes

**Description**

The getJobInfo() function gets information on a job from its Id

**Return Value**

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
VISHNU_OK	The service was performed successfully.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

**Signature**

```
int getJobInfo(const string& sessionKey, const string& machineId, const string& jobId, Job& jobInfos);
```

**3.1.4 Function TMS : :cancelJob****Access**

This function can be used by any Vishnu user.

**Parameters**



Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
jobId	string	The Id of the job	IN	yes
infoMsg	string	The information message	OUT	yes

### Description

The cancelJob() function cancels a job from it Id

### Return Value

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
VISHNU_OK	The service was performed successfully.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

### Signature

```
int cancelJob(const string& sessionKey, const string& machineId, const string& jobId, string& infoMsg);
```

## 3.1.5 Function TMS : :getJobOutPut

### Access

This function can be used by any Vishnu user.

### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
jobId	string	The Id of the job	IN	yes
outputPath	string	The path of the file containing the output result of the job	OUT	yes
errorPath	string	The path of the file containing the errors that has been occurred during the execution of the job	OUT	yes

### Description

The getJobOutPut() function gets outputPath and errorPath of a job from it Id

### Return Value

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
VISHNU_OK	The service was performed successfully.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

### Signature

int **getJobOutPut**(const string& sessionKey, const string& machineId, const string& jobId, string& outputPath, string& errorPath);

### 3.1.6 Function TMS : :getAllJobsOutPut

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
listOfResults	ListJobResults	Is the list of jobs results	OUT	yes

#### Description

The getAllJobsOutPut() function gets outputPath and errorPath of completed jobs dynamically

#### Return Value

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
VISHNU_OK	The service was performed successfully.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

### Signature

int **getAllJobsOutPut**(const string& sessionKey, const string& machineId, ListJobResults& listOfResults);

### 3.1.7 Function TMS : :listQueues

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Machine hash key	IN	yes
listofQueues	ListQueues	The list of queues	OUT	yes

#### Description

The listQueues() function gets queues information

#### Return Value

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

Name	Description
TMS_BATCH_SCHEDULER_ERROR	Indicates an error caused by the underlying batch scheduler
TMS_INVALID_RESPONSE	Indicates that the implementation produced a response that does not match the criteria defined by the specification.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_PERMISSION_DENIED	Indicates the requested operation is not allowed for provided user.
VISHNU_OK	The service was performed successfully.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_UNKNOWN_BATCH_SCHEDULER_TYPE	Indicates that the batch scheduler type is not known.
TMS_UNKNOWN_MACHINE	The machine is not known.

#### Signature

```
int listQueues(const string& sessionKey, const string& machineId, ListQueues& listofQueues);
```

### 3.1.8 Function TMS : :setMachineEnv

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Represents the machine id	IN	yes
listEnv	string	Represents the list environnement variables	IN	yes

#### Description

The setMachineEnv() function sets environment variables on a remote machine

#### Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully.
TMS_UNKNOWN_MACHINE	The machine is not known.

**Signature**

```
int setMachineEnv(const string& sessionKey, const string& machineId, const string& listEnv);
```

**3.1.9 Function TMS : :setMachineRefreshPeriod****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	The id of the machine	IN	yes
value	int	Is the refresh interval value (in seconds)	IN	yes

**Description**

The setMachineRefreshPeriod() function sets the refresh period of output and error file content

**Return Value**

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

Name	Description
VISHNU_OK	The service was performed successfully.
TMS_UNKNOWN_MACHINE	The machine is not known.

**Signature**

```
int setMachineRefreshPeriod(const string& sessionKey, const string& machineId, const int& value);
```

**3.1.10 Function TMS : :getJobProgress****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The sessionKey is the identifier of the session generated by VISHNU	IN	yes
machineId	string	Is the id of the machine to get the jobs progression.	IN	yes
progress	Progression	Is the object containing jobs progression information	OUT	yes
options	ProgressOptions	Is an object containing the available options jobs for progression .	IN	no

**Description**

The getJobProgress() function get the progression status of jobs.

## Return Value

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

Name	Description
VISHNU_OK	The service was performed successfully.
TMS_INVALID_REQUEST	Indicates that the request is not valid.
TMS_UNKNOWN_MACHINE	The machine is not known.
TMS_INVALID_SESSION_KEY	The session key is not valid to perform the service.
TMS_SERVER_NOT_AVAILABLE	Indicates that the task management server is not available.
TMS_SUBMIT_SERVICE_NOT_AVAILABLE	Indicates that the service to perform the submit operation is not found.

## Signature

```
int getJobProgress(const string& sessionKey, const string& machineId, Progression& progress, const ProgressOptions& options=ProgressOptions());
```

## 3.2 Data types definitions

### SubmitOptions Class Content

Name	Type	Description
name	string	Assigns a job name. The default is the name of job path.
priority	JobPriority	Assigns priority of the job.
queue	string	Assigns the queue or class of the job.
wallTime	int	The maximum wall-clock time during which the job can run.
memory	int	The size of memory that the job will use.
nbCpu	int	The number of cpu that the job will use.
nbNodesAndCpuPerNode	int	The number of nodes and processors per node.
OutPutPath	string	Assigns the path and file for job output.
ErrorPath	string	Assigns the path and file for job error.

### JobPriority Enumeration Type

Name	Value
VERY_LOW	100
LOW	200
NORMAL	300
HIGH	400
VERY_HIGH	500

### ListJobs Class Content

Name	Type	Description
nbJobs	long	Represents the total number of jobs in the list.
nbRunningJobs	long	Represents of running jobs in the list.
nbWaitingJobs	long	Represents the total number of waiting jobs in the list.
jobs	List of Job	Is a list of job information (jobId, jobName, ...).

### Job Class Content

Name	Type	Description
sessionId	string	Is the id of the session that contained the job submission command
submitMachineId	string	Is the id of the machine on which the job has been submitted.
submitMachineName	string	Is the name of the machine on which the job has been submitted.
jobId	string	Represents the id to job.
jobName	string	Represents the name assigned to the job.
jobPath	string	Is the path to the file containing job characteristics.
outputPath	string	Is the path to the job output results.
errorPath	string	Is the path to the file containing errors occurred during job's execution.
jobPrio	JobPriority	Represents the job priority.
nbCpus	int	Is the number of cpu used by the job.
jobWorkingDir	string	Indicates the directory where the job has been launched.
status	JobStatus	The current status of the job.
submitDate	long	Date and time when job was submitted (unix timestamp)
endDate	long	Represents the execution end date of the job (unix timestamp)
owner	string	Represents the job owner.
jobQueue	string	Is the name of the queue or class associated to the job.
wallClockLimit	long	Is the maximum wall-clock time during which the job can run (in seconds)
groupName	string	Represents the job owner group name.
jobDescription	string	Is the textual description of the job.
memLimit	int	Represents the memory size limit of the job.
nbNodes	int	Is the total number of nodes used by the job.
nbNodesAndCpuPerNode	int	Is the number of nodes and processors per node used by the job.

### JobStatus Enumeration Type

Name	Value
RUNNING	0
WAITING	1
COMPLETED	2
CANCELED	3
HELD	4
QUEUED	5
FAILED	6
NOT_SUBMITTED	7

### ListJobsOptions Class Content

Name	Type	Description
JobId	string	To list job which has this id.
nbCpu	int	To list jobs which have this number of cpu.
fromSubmitDate	long	List jobs submitted after this date (unix timestamp).
toSubmitDate	long	List jobs submitted before this date (unix timestamp)
owner	string	To list all jobs submitted by this owner.
status	JobStatus	To list jobs which have this status.
priority	JobPriority	To list jobs which have this priority
OutPutPath	string	Gets the path and file for each job output.
ErrorPath	string	Gets the path and file for each job error.

Name	Type	Description
queue	string	To list jobs which have this queue name.

#### ListJobResults Class Content

Name	Type	Description
nbJobs	string	Is the number of jobs.
Results	List of <b>JobResult</b>	Represents the list of completed jobs results.

#### JobResult Class Content

Name	Type	Description
jobId	string	Represents the id of the job.
outputPath	string	Is the path to the job output results.
errorPath	string	Is the path to the file containing errors occurred during job's execution.

#### ListQueues Class Content

Name	Type	Description
nbQueues	int	Represents the number of queues.
queues	List of <b>Queue</b>	Represents the list of queues.

#### Queue Class Content

Name	Type	Description
name	string	Is the queue name.
maxJobCpu	int	Is the maximum number of Cpus that a job can use.
maxProcCpu	int	Is the maximum number of Cpus of the queue.
memory	int	Represents the queue memory size.
wallTime	long	Is the total wallTime of the queue.
node	int	Is the maximum number of nodes of the queue.
nbRunningJobs	int	Is the total running jobs in the queue.
nbJobsInQueue	int	Is the total number of jobs in the queue.
state	<b>QueueStatus</b>	Is the status of the queue.
priority	<b>QueuePriority</b>	Represents the priority of the queue.
description	string	Is the queue description.

#### QueueStatus Enumeration Type

Name	Value
STARTED	0
RUNNING	1
NOT_STARTED	2
NOT_AVAILABLE	3

#### QueuePriority Enumeration Type

Name	Value
VERY_LOW	0
LOW	1
NORMAL	2
HIGH	3



Name	Value
VERY_HIGH	4

**Progression Class Content**

Name	Type	Description
jobId	string	Represents the job id.
jobName	string	Represents the job name.
wallTime	int	Represents the job wall time.
startTime	long	Start date and time of the job (unix timestamp)
endTime	long	End date and time of the job (unix timestamp)
percent	double	Represent the job progression.
status	<b>JobStatus</b>	Represents the job status.

**ProgressOptions Class Content**

Name	Type	Description
jobId	string	Represents the id of the job that the user wants to see its progression.
jobOwner	string	Represents the owner of the job.



## Chapitre 4

# API specification for Information Management System (IMS)

### 4.1 Definition of the functions of the package

#### 4.1.1 Function IMS : :getUpdateFrequency

##### Access

This function can be used by any Vishnu user.

##### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
freq	int	Frequency the data are updated, in second	OUT	yes

##### Description

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

##### Signature

```
int getUpdateFrequency(const string& sessionKey, int& freq);
```

#### 4.1.2 Function IMS : :export

##### Access

This function can be used by any Vishnu user.

##### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command (current session key)	IN	yes
oldSessionKey	string	The key of the session to export (session has ended)	IN	yes
exportType	ExportType	The format to export	IN	yes
filename	string	The file containing the commands	OUT	yes

### Description

The export() 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

### Signature

```
int export(const string& sessionKey, const string& oldSessionKey, const ExportType& exportType, string& filename);
```

## 4.1.3 Function IMS : :replay

### Access

This function can be used by any Vishnu user.

### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
filename	string	A file containing the commands to replay	IN	yes
type	ExportType	The type of the script to execute (python, shell)	IN	yes

### Description

The replay() function replays a script of VISHNU commands

### Return Value

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

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

### Signature

```
int replay(const string& sessionKey, const string& filename, const ExportType& type);
```

## 4.1.4 Function IMS : :getMetricVal

### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine	IN	yes
startTime	long	The start time to look for the metric	IN	yes
endTime	long	The end time to search	IN	yes
metricType	MetricType	The metric types	IN	yes
res	ListMetric	A list of corresponding results	OUT	yes

#### Description

The getMetricVal() function gets data from a metric

#### Return Value

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

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

#### Signature

```
int getMetricVal(const string& sessionKey, const string& machineId, const long& startTime, const long& endTime, const MetricType& metricType, ListMetric& res);
```

### 4.1.5 Function IMS : :getCurrentData

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine the user wants the total diskSpace	IN	yes
dataType	MetricType	The type of data the user wants to get	IN	yes
res	double	The total diskSpace on the machine	OUT	yes

#### Description

The getCurrentData() function gets data corresponding to the datatype on a machine

#### Return Value

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

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int getCurrentData(const string& sessionKey, const string& machineId, const MetricType& dataType, double& res);
```

**4.1.6 Function IMS : :getProcesses****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine the user wants the running processes	IN	yes
process	ListProcesses	The list of the processes on the machine	OUT	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int getProcesses(const string& sessionKey, const string& machineId, ListProcesses& process);
```

**4.1.7 Function IMS : :setSystemThreshold****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine the user wants to set the threshold over	IN	yes
thresholdType	MetricType	The type of the metric to set	IN	yes
value	double	The threshold value	IN	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int setSystemThreshold(const string& sessionKey, const string& machineId, const MetricType& tresholdType, const double& value);
```

**4.1.8 Function IMS : :getSystemThreshold****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine the user wants to get the treshold over	IN	yes
type	MetricType	The treshold type desired	IN	yes
value	double	The treshold value	OUT	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int getSystemThreshold(const string& sessionKey, const string& machineId, const MetricType& type, double& value);
```

**4.1.9 Function IMS : :defineUserIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
format	string	The new format to use	IN	yes

**Description**

The `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 function.

Name	Description
VISHNU_OK	Error code returned if success
INVALID_PARAMETER	If a parameter is invalid
DBERROR	The database generated an error

#### Signature

```
int defineUserIdentifier(const string& sessionKey, const string& format);
```

### 4.1.10 Function IMS : `defineMachineIdentifier`

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
format	string	The new format to use	IN	yes

#### Description

The `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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

#### Signature

```
int defineMachineIdentifier(const string& sessionKey, const string& format);
```

### 4.1.11 Function IMS : `defineJobIdentifier`

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
format	string	The new format to use	IN	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int defineJobIdentifier(const string& sessionKey, const string& format);
```

**4.1.12 Function IMS : :defineTransferIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
format	string	The new format to use	IN	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int defineTransferIdentifier(const string& sessionKey, const string& format);
```

**4.1.13 Function IMS : :loadShed****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine to stop	IN	yes
loadShedType	LoadShedType	Type of load shedding	IN	yes

**Description**

The loadShed() function loads shed a machine, 2 modes are available, soft or hard. Soft flushes the machine, hard stops the machine

**Return Value**

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

Name	Description
VISHNU_OK	Error code returned if success
COMPONENT_ERROR	If a component is unavailable
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int loadShed(const string& sessionKey, const string& machineId, const LoadShedType& loadShedType);
```

**4.1.14 Function IMS : :setUpdateFrequency****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
freq	int	Frequency the data are updated, in second	IN	yes

**Description**

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

**Signature**

```
int setUpdateFrequency(const string& sessionKey, const int& freq);
```

**4.1.15 Function IMS : :restart****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes



Parameter	Type	Description	Mode	Required
type	<b>RestartType</b>	The type of restart desired	IN	yes
componentName	string	If not restarting all, the name of the component to restart	IN	no

### Description

The 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 function.

Name	Description
VISHNU_OK	Error code returned if success
COMPONENT_ERROR	If a component is unavailable
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

### Signature

```
int restart(const string& sessionKey, const RestartType& type, const string& componentName);
```

## 4.1.16 Function IMS : :updateMachine

### Access

This function can be used by ADMIN users only

### Parameters

Parameter	Type	Description	Mode	Required
sessionKey	string	The key of the session of the user that submits the command	IN	yes
machineId	string	The id of the machine to update	IN	yes
CompoMachine	<b>CompoMachine</b>	A description of the type of component to change	IN	yes
val	<b>object</b>	The value to set	IN	yes
lang	string	For machine description, the language used	IN	no

### Description

The updateMachine() function updates the machines information in the database (description, disk space or memory)

### Return Value

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

Name	Description
VISHNU_OK	Error code returned if success
DBERROR	The database generated an error
INVALID_PARAMETER	If a parameter is invalid

### Signature

```
int updateMachine(const string& sessionKey, const string& machineId, const CompoMachine& CompoMachine, const object& val, const string& lang);
```

## 4.2 Data types definitions

### ExportType Enumeration Type

Name	Value
PYTHON	0
SHELL	1

### MetricType Enumeration Type

Name	Value
CPUNBR	0
CPUUSE	1
DISKSPACE	2
FREEDISKSPACE	3
MEMORY	4
FREEMEMORY	5

### ListMetric Class Content

Name	Type	Description
metric	List of <b>Metric</b>	The metrics of the list

### Metric Class Content

Name	Type	Description
type	<b>MetricType</b>	The type of the metric
value	double	The value of the metric
time	int	The timestamp the metric had the value

### ListProcesses Class Content

Name	Type	Description
processName	List of string	The processes of the list

### LoadShedType Enumeration Type

Name	Value
SOFT	0
HARD	1

### RestartType Enumeration Type

Name	Value
ALL	0
SED	1
AGENT	2
DAEMON	3
MACHINE	4

### CompoMachine Enumeration Type

Name	Value
DESC	0
DISKSPACE	1
RAM	2

## Chapitre 5

# API specification for File Management System (FMS)

### 5.1 Definition of the functions of the package

#### 5.1.1 Function FMS : :createFile

##### Access

This function can be used by any Vishnu user.

##### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
pathList	<b>StringList</b>	The list of files to create, each path following the pattern [host :]file path.	IN	yes
mode	long	the file access permissions.	IN	no

##### Description

The createFile() function creates files on remote machines.

##### Return Value

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unkown error occured.

##### Signature

```
int createFile(const string& machineId, const string& sessionKey, const StringList& pathList, const long& mode);
```

### 5.1.2 Function FMS : :createDir

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
pathList	StringList	The list of directories to create, each path following the pattern [host :]file path.	IN	yes
mode	long	the new directories permission access.	IN	no

#### Description

The createDir() function creates directories on remote machines.

#### Return Value

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

Name	Description
VISHNU_OK	The service has been performed successfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unknown error occurred.

#### Signature

```
int createDir(const string& machineId, const string& sessionKey, const StringList& pathList, const long& mode);
```

### 5.1.3 Function FMS : :removeFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	The list of files to delete, each path following the pattern [host :]file path.	IN	yes

#### Description

The removeFile() function removes files from remote hosts.

**Return Value**

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unknwn error occured.

**Signature**

```
int removeFile(const string& machineId, const string& sessionKey, const StringList& pathList);
```

**5.1.4 Function FMS : :removeDir****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	<b>StringList</b>	The list of directories to delete, each path following the pattern [host :]directory path.	IN	yes

**Description**

The removeDir() function removes directories (and subdirectories ) from remote machines.

**Return Value**

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unknwn error occured.

**Signature**

```
int removeDir(const string& machineId, const string& sessionKey, const StringList& pathList);
```

### 5.1.5 Function FMS : :chGrpOfFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	The list of files , each path following the pattern [host :]file path.	IN	yes
group	string	the new group owner of files.	IN	yes

#### Description

The chGrpOfFile() function changes group owner of remote files.

#### Return Value

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unkown error occured.

#### Signature

```
int chGrpOfFile(const string& machineId, const string& sessionKey, const StringList& pathList, const string& group);
```

### 5.1.6 Function FMS : :chModOfFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	The list of files, each path following the pattern [host :]file path.	IN	yes
mode	long	the access rigths of a file	IN	yes

#### Description

The chModOfFile() function changes access rights of remote files.

#### Return Value

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
UNKNOWN_ERROR	an unkown error occured.

### Signature

```
int chModOfFile(const string& machineId, const string& sessionKey, const StringList& pathList, const long& mode);
```

### 5.1.7 Function FMS : :headOfFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	<b>StringList</b>	The list of files , each path following the pattern [host :]file path.	IN	yes
nLine	int	number of lines to display	IN	no
buffer	<b>StringList</b>	a buffer in which the first "nline" lines will be stored. he string buffer[i] represents the last "nline" of the file located in pathList[i] (0<=i<length(pathList)).	OUT	yes

### Description

The headOfFile() function displays a few first lines of files located on remote machines.

### Return Value

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

Name	Description
VISHNU_OK	The service has been performed succesfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unkown error occured.

### Signature



```
int headOfFile(const string& machineId, const string& sessionKey, const StringList& pathList, const int& nLine, StringList& buffer);
```

### 5.1.8 Function FMS : :tailOfFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	The path of the file to create as [host :]file path.	IN	yes
nLine	int	number of lines to display	IN	no
buffer	StringList	a buffer in which the last "nLine" lines will be stored. The string buffer[i] represents the last "nline" of the file located in pathList[i] (0<=i<length(pathList)).	OUT	yes

#### Description

The tailOfFile() function displays a few last lines of files located on remote machines

#### Return Value

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

Name	Description
VISHNU_OK	The service has been performed successfully.
INVALID_SESSION_KEY	Invalid provided session key.
INVALID_MACHINE_ID	Invalid provided machine identifier.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
FMS_PERMISSION_DENIED	User does not have permission access.
DB_ERROR	A problem occurs with the database.
UNKNOWN_ERROR	an unknown error occurred.

#### Signature

```
int tailOfFile(const string& machineId, const string& sessionKey, const StringList& pathList, const int& nLine, StringList& buffer);
```

### 5.1.9 Function FMS : :listDir

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	The list of directories, each path following the pattern [host :]directory path.	IN	yes

Parameter	Type	Description	Mode	Required
buffer	StringList	a buffer in which the contents of the directories will be stored.	OUT	yes
options	LsDirOptions		IN	yes

### Description

The listDir() function displays the contents of remote directory.

### Return Value

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

### Signature

```
int listDir(const string& machineId, const string& sessionKey, const StringList& pathList, StringList& buffer, const LsDirOptions& options=LsDirOptions());
```

## 5.1.10 Function FMS : :copyFile

### Access

This function can be used by any Vishnu user.

### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
src	StringList	The list of source files to copy, each path following the pattern [host :]file path.	IN	yes
dest	string	The path of the destination file	IN	yes
options	CopyFileOptions	the copy options.	IN	no

### Description

The copyFile() function executes a synchronous copy of file.

### Return Value

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.

Name	Description
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

### Signature

```
int copyFile(const string& machineId, const string& sessionKey, const StringList& src, const string& dest, const CopyFileOptions& options=CopyFileOptions());
```

### 5.1.11 Function FMS : :copyAsyncFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
src	StringList	The list of source files to copy, each path following the pattern [host :]file path.	IN	yes
dest	string	The path of the destination file	IN	yes
options	CopyFileOptions	the copy options.	IN	no
thrRef	LongList	a list of file transfer identifiers	OUT	yes

#### Description

The copyAsyncFile() function executes an asynchronous copy of file.

#### Return Value

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

### Signature

```
int copyAsyncFile(const string& machineId, const string& sessionKey, const StringList& src, const string& dest, const CopyFileOptions& options=CopyFileOptions(), LongList& thrRef);
```

### 5.1.12 Function FMS : :moveFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
src	StringList	The list of source files to move, each path following the pattern [host :]file path.	IN	yes
dest	string	The path of the destination file	IN	yes
options	MvFileOptions	the move command options.	IN	no

#### Description

The moveFile() function executes a synchronous move of file.

#### Return Value

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unkown error occured.
VISHNU_OK	The service has been performed succesfully.

#### Signature

```
int moveFile(const string& machineId, const string& sessionKey, const StringList& src, const string& dest, const MvFileOptions& options=MvFileOptions());
```

### 5.1.13 Function FMS : :moveAsyncFile

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
src	StringList	The list of source files to move, each path following the pattern [host :]file path.	IN	yes
dest	string	The path of the destination file	IN	yes
thrRef	LongList	a list of file tranfer identifiers	OUT	yes
options	MvFileOptions	the mv command options.	IN	no

**Description**

The moveAsyncFile() function executes an asynchronous move of file.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int moveAsyncFile(const string& machineId, const string& sessionKey, const StringList& src, const string& dest, LongList& thrRef, const MvFileOptions& options=MvFileOptions());
```

**5.1.14 Function FMS : :stopFileTransfer****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
thrRef	LongList	a list of file tranfer identifiers	IN	yes

**Description**

The stopFileTransfer() function stops an execution of an set of file transfer.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int stopFileTransfer(const string& machineId, const string& sessionKey, const LongList& thrRef);
```

**5.1.15 Function FMS : :stopAllFileTransfer****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
options	LsFileTransferOptions		IN	yes

**Description**

The stopAllFileTransfer() function stops all file transfer in a session.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int stopAllFileTransfer(const LsFileTransferOptions& options=LsFileTransferOptions());
```

**5.1.16 Function FMS : :stopAllUsersFileTransfer****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
options	LsFileTransferOptions		IN	yes

**Description**

The stopAllUsersFileTransfer() function stops all users file transfer.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.

Name	Description
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int stopAllUsersFileTransfer(const LsFileTransferOptions& options=LsFileTransferOptions());
```

**5.1.17 Function FMS : :getFileTransferStatus****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session identifier	IN	yes
thrRef	LongList	a list of file transfer identifiers	IN	yes

**Description**

The getFileTransferStatus() function gives the status of a list of file transfer.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int getFileTransferStatus(const string& machineId, const string& sessionKey, const LongList& thrRef);
```

**5.1.18 Function FMS : :listFilesTransferStatus****Access**

This function can be used by any Vishnu user.



**Parameters**

Parameter	Type	Description	Mode	Required
options	LsFileTransferOptions		IN	yes

**Description**

The listFilesTransferStatus() function displays the status of all files transfer submitted by User.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unkown error occured.
VISHNU_OK	The service has been performed succesfully.

**Signature**

```
int listFilesTransferStatus(const LsFileTransferOptions& options=LsFileTransferOptions());
```

**5.1.19 Function FMS : :listAllUsersFilesTransferStatus****Access**

This function can be used by ADMIN users only

**Parameters**

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
options	LsFileTransferOptions		IN	yes

**Description**

The listAllUsersFilesTransferStatus() function displays the status of all files transfer submitted by User.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.



Name	Description
UNKNOWN_ERROR	an unkown error occured.
VISHNU_OK	The service has been performed succesfully.

### Signature

```
int listAllUsersFilesTransferStatus(const string& machineId, const string& sessionKey, const LsFileTransferOptions& options=LsFileTransferOptions());
```

### 5.1.20 Function FMS : :DisplayFilesTransferHistory

#### Access

This function can be used by any Vishnu user.

#### Parameters

Parameter	Type	Description	Mode	Required
options	LsFileTransferOptions		IN	yes

#### Description

The DisplayFilesTransferHistory() function displays the history of all files transfer submitted by User.

#### Return Value

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unkown error occured.
VISHNU_OK	The service has been performed succesfully.

### Signature

```
int DisplayFilesTransferHistory(const LsFileTransferOptions& options=LsFileTransferOptions());
```

### 5.1.21 Function FMS : :DisplayAllUsersFilesTransferHistory

#### Access

This function can be used by ADMIN users only

#### Parameters

Parameter	Type	Description	Mode	Required
options	LsFileTransferOptions		IN	yes

#### Description

The DisplayAllUsersFilesTransferHistory() function displays the history of all files transfer submitted by all Users.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int DisplayAllUsersFilesTransferHistory(const LsFileTransferOptions& options=LsFileTransferOptions());
```

**5.1.22 Function FMS : :GetFilesInfo****Access**

This function can be used by any Vishnu user.

**Parameters**

Parameter	Type	Description	Mode	Required
machineId	string	The machine identifier	IN	yes
sessionKey	string	The session id	IN	yes
pathList	StringList	the list of file whose inode information will be displayed.	IN	yes
filesinfo	FileStatList	the list of inode.	OUT	yes

**Description**

The GetFilesInfo() function displays the information of files.

**Return Value**

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

Name	Description
DB_ERROR	A problem occurs with the database.
FMS_INVALID_CFG_FILE	Option unknown in the config file.
FMS_INVALID_OPTION	The option provided is invalid for this service.
FMS_INVALID_PATH	The path provided is invalid.
FMS_PERMISSION_DENIED	User does not have permission access.
FMS_SERVER_UNAVAILABLE	The FMS server is unavailable.
INVALID_MACHINE_ID	Invalid provided machine identifier.
INVALID_SESSION_KEY	Invalid provided session key.
UNKNOWN_ERROR	an unknown error occurred.
VISHNU_OK	The service has been performed successfully.

**Signature**

```
int GetFilesInfo(const string& machineId, const string& sessionKey, const StringList& pathList, FileStatList& filesinfo);
```

## 5.2 Data types definitions

### StringList Class Content

Name	Type	Description
Error	Error	TO DO
listOfStrings	List of string	the dynamic list of strings.

### LsDirOptions Class Content

Name	Type	Description
Error	Error	TO DO
isRecursive	boolean	TO DO

### CopyFileOptions Class Content

Name	Type	Description
Error	Error	TO DO
isRecursive	boolean	TO DO
transferCommand	<b>TransferCommand</b>	the command to use to perform file transfer.

### TransferCommand Enumeration Type

Name	Value
SCP	0
RSYNC	1

### LongList Class Content

Name	Type	Description
Error	Error	TO DO
listOfLongs	List of long	TO DO

### MvFileOptions Class Content

Name	Type	Description
Error	Error	TO DO
transferCommand	<b>TransferCommand</b>	the command to use to perform file transfer.

### LsFileTransferOptions Class Content

Name	Type	Description
Error	Error	TO DO
sessionKey	string	the command to use to perform file transfer.
machineId	string	the command to use to perform file transfer.
nbFileTransfers	int	the command to use to perform file transfer.
status	<b>Status</b>	the command to use to perform file transfer.
userId	string	the user identifier

### Status Enumeration Type

Name	Value
INPROGRESS	0
COMPLETED	1
CANCELLED	2
FAILED	3

**FileStatList Class Content**

Name	Type	Description
listOfFileStat	List of <b>FileStat</b>	list of inodes.

**FileStat Class Content**

Name	Type	Description
Error	Error	TO DO
path	string	TO DO
owner	string	TO DO
group	string	TO DO
uid	long	TO DO
gid	long	TO DO
size	long	TO DO
atime	long	TO DO
mtime	long	TO DO
ctime	long	TO DO
type	<b>FileType</b>	TO DO

**FileType Enumeration Type**

Name	Value
BLOCK	0
CHARACTER	1
DIRECTORY	2
SYMBOLICLINK	3
SCKT	4
FIFO	5
REGULAR	6