

## **D1.1c - VISHNU A.P.I. specifications**



## COLLABORATORS

|               | <i>TITLE :</i><br>D1.1c - VISHNU A.P.I. specifications   |               |                  |
|---------------|--|---------------|------------------|
| <i>ACTION</i> | <i>NAME</i>  | <i>DATE</i>   | <i>SIGNATURE</i> |
| WRITTEN BY    | Benjamin Isnard,<br>Daouda Traoré,<br>Eugène Pamba<br>Capo-Chichi, Kevin<br>Coulomb, and<br>Ibrahima Cissé | June 30, 2011 |                  |

## REVISION HISTORY

| NUMBER | DATE       | DESCRIPTION   | NAME    |
|--------|------------|---|---------|
| 0      | 05/01/2011 | Formatting example  | SysFera |
| 1      | 13/01/2011 | First release   | SysFera |
| 2      | 10/02/2011 | Removed UMS::AddUserOptions,<br>UMS::AddMachineOptions and<br>UMS::UpdateUserOptions classes and<br>UMS::SessionStateType.<br>Added function UMS::reconnect,<br>UMS::StatusType.<br>Modified UMS::restoreConfiguration class      | SysFera |
| 3      | 03/03/2011 | Renamed UMS::AddVishnuUser, Modified the<br>signature of UMS::connect, UMS::reconnect,<br>UMS::resetPassword, UMS::addMachine,<br>UMS::addLocalAccount, UMS::saveConfiguration<br>and IMS::exportCommands<br>Updated error codes. | SysFera |
| 4      | 30/03/2011 | Modified signature FMS::listDir and<br>FMS::ListFileTransfers. Removed<br>FMS::ListFileFileTransferStatus. Added data type<br>FMS::FileTransfer, FMS::FileTransferList,<br>FMS::FileStatList.                                     | SysFera |

**REVISION HISTORY**

| NUMBER | DATE       | DESCRIPTION   | NAME    |
|--------|------------|---|---------|
| 5      | 15/04/2011 | <p>Updates to TMS: Modified description of machineId parameter. Modified getJobProgress output parameter. Modified submitJob description and output parameters (jobId and jobPath replaced by jobInfo). Modified name of getAllJobsOutput to getCompletedJobsOutput. Added input parameter outDir to getJobOutput and getCompletedJobsOutput. Modified output parameter of getCompletedJobsOutput. Added service getMachineRefreshPeriod. Removed function setMachineEnv. Removed JobPriority attribute in submitOptions class. Removed Not_submitted status value of Job class. Modified values of JobStatus Enum. Updated error messages and codes.</p> <p>Updates to IMS : Add option in exportCommand, getMetricCurrentValue, getProcesses, getMetricHistory, restart and getSystemThreshold. Add services stop and getSystemInfo. Add corresponding datatypes (option) plus list of system info and processes.</p> | SysFera |
| 6      | 21/04/2011 | <p>Updates to TMS: Removed functions getMachineRefreshPeriod and setMachineRefreshPeriod. Added UNDEFINED and ALREADY_DOWNLOADED status of JobStatus Enum. Added UNDEFINED priority of JobPriority Enum. Removed of outputPah and errorPath of ListJobsOptions. Removed scriptContent in Job class. Added ERRCODE_ALREADY_TERMINATED, ERRCODE_ALREADY_CANCELED, ERRCODE_JOB_IS_NOT_TERMINATED, ERRCODE_ALREADY_DOWNLOADED in exception list. Modified values of JobPriority Enum. Modified values of QueuePriority and QueueStatus. Added UNDEFINED priority of QueuePriority.</p>  | SysFera |
| 7      | 26/04/2011 | <p>Updates of UMS: Added CommandStatusType Enum and an attribute status to the Command Class.</p>   | SysFera |
| 8      | 24/05/2011 | <p>Updates of UMS: Added new error code for addLocalAccount and updateLocalAccount.</p>   | SysFera |

**REVISION HISTORY**

| NUMBER | DATE       | DESCRIPTION   | NAME    |
|--------|------------|---|---------|
| 9      | 29/06/2011 | Updates of UMS: Modified API functions names listLocalAccount and listMachine to listLocalAccounts and listMachines.<br>Updates of FMS: Modified API function name getFilesInfo to getFileInfo. | SysFera |

# Contents

|          |  |          |
|----------|--|----------|
| <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 . . . . .                        | 2        |
| 1.3.1.1  | Generic method definition format . . . . .                 | 2        |
| 1.3.1.2  | C++ specific aspects . . . . .                             | 2        |
| 1.3.1.3  | Python specific aspects . . . . .                          | 2        |
| 1.3.1.4  | Web Services specific aspects . . . . .                    | 3        |
| 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  | Python specific aspects . . . . .                          | 3        |
| 1.3.2.4  | Web Services specific aspects . . . . .                    | 3        |
| 1.4      | Web Services description . . . . .                         | 3        |
| 1.5      | References . . . . .                                       | 4        |
| 1.6      | Glossary . . . . .   | 4        |
| <b>2</b> | <b>API specification for User Management Service (UMS)</b> | <b>5</b> |
| 2.1      | Definition of the functions of the package . . . . .       | 5        |
| 2.1.1    | Function UMS::connect . . . . .                            | 5        |
| 2.1.2    | Function UMS::reconnect . . . . .                          | 6        |
| 2.1.3    | Function UMS::addUser . . . . .                            | 6        |
| 2.1.4    | Function UMS::updateUser . . . . .                         | 8        |
| 2.1.5    | Function UMS::deleteUser . . . . .                         | 9        |
| 2.1.6    | Function UMS::close . . . . .                              | 10       |
| 2.1.7    | Function UMS::changePassword . . . . .                     | 10       |
| 2.1.8    | Function UMS::resetPassword . . . . .                      | 11       |
| 2.1.9    | Function UMS::addLocalAccount . . . . .                    | 11       |
| 2.1.10   | Function UMS::updateLocalAccount . . . . .                 | 12       |

|          |   |           |
|----------|---|-----------|
| 2.1.11   | Function UMS::deleteLocalAccount . . . . .                        | 13        |
| 2.1.12   | Function UMS::saveConfiguration . . . . .                         | 14        |
| 2.1.13   | Function UMS::restoreConfiguration . . . . .                      | 14        |
| 2.1.14   | Function UMS::addMachine . . . . .                                | 15        |
| 2.1.15   | Function UMS::updateMachine . . . . .                             | 16        |
| 2.1.16   | Function UMS::deleteMachine . . . . .                             | 16        |
| 2.1.17   | Function UMS::listLocalAccounts . . . . .                         | 17        |
| 2.1.18   | Function UMS::listMachines . . . . .                              | 18        |
| 2.1.19   | Function UMS::listHistoryCmd . . . . .                            | 18        |
| 2.1.20   | Function UMS::listOptions . . . . .                               | 19        |
| 2.1.21   | Function UMS::listUsers . . . . .                                 | 20        |
| 2.1.22   | Function UMS::listSessions . . . . .                              | 21        |
| 2.1.23   | Function UMS::configureDefaultOption . . . . .                    | 21        |
| 2.1.24   | Function UMS::configureOption . . . . .                           | 22        |
| 2.1.25   | Function UMS::vishnuInitialize . . . . .                          | 23        |
| 2.1.26   | Function UMS::vishnuFinalize . . . . .                            | 23        |
| 2.2      | Data types definitions . . . . .                                  | 24        |
| <b>3</b> | <b>API specification for Tasks Management Service (TMS)</b>       | <b>29</b> |
| 3.1      | Definition of the functions of the package . . . . .              | 29        |
| 3.1.1    | Function TMS::submitJob . . . . .                                 | 29        |
| 3.1.2    | Function TMS::getJobInfo . . . . .                                | 30        |
| 3.1.3    | Function TMS::getJobProgress . . . . .                            | 31        |
| 3.1.4    | Function TMS::listQueues . . . . .                                | 31        |
| 3.1.5    | Function TMS::listJobs . . . . .                                  | 32        |
| 3.1.6    | Function TMS::getJobOutput . . . . .                              | 33        |
| 3.1.7    | Function TMS::getCompletedJobsOutput . . . . .                    | 33        |
| 3.1.8    | Function TMS::cancelJob . . . . .                                 | 34        |
| 3.2      | Data types definitions . . . . .                                  | 35        |
| <b>4</b> | <b>API specification for Information Management Service (IMS)</b> | <b>39</b> |
| 4.1      | Definition of the functions of the package . . . . .              | 39        |
| 4.1.1    | Function IMS::exportCommands . . . . .                            | 39        |
| 4.1.2    | Function IMS::getMetricCurrentValue . . . . .                     | 40        |
| 4.1.3    | Function IMS::getMetricHistory . . . . .                          | 40        |
| 4.1.4    | Function IMS::getProcesses . . . . .                              | 41        |
| 4.1.5    | Function IMS::setSystemInfo . . . . .                             | 41        |
| 4.1.6    | Function IMS::setSystemThreshold . . . . .                        | 42        |
| 4.1.7    | Function IMS::getSystemThreshold . . . . .                        | 42        |

|          |  |           |
|----------|--|-----------|
| 4.1.8    | Function IMS::defineUserIdentifier . . . . .               | 43        |
| 4.1.9    | Function IMS::defineMachineIdentifier . . . . .            | 43        |
| 4.1.10   | Function IMS::defineJobIdentifier . . . . .                | 44        |
| 4.1.11   | Function IMS::defineTransferIdentifier . . . . .           | 44        |
| 4.1.12   | Function IMS::loadShed . . . . .                           | 45        |
| 4.1.13   | Function IMS::setUpdateFrequency . . . . .                 | 45        |
| 4.1.14   | Function IMS::getUpdateFrequency . . . . .                 | 46        |
| 4.1.15   | Function IMS::stop . . . . .                               | 46        |
| 4.1.16   | Function IMS::getSystemInfo . . . . .                      | 47        |
| 4.1.17   | Function IMS::restart . . . . .                            | 47        |
| 4.2      | Data types definitions . . . . .                           | 48        |
| <b>5</b> | <b>API specification for File Management Service (FMS)</b> | <b>51</b> |
| 5.1      | Definition of the functions of the package . . . . .       | 51        |
| 5.1.1    | Function FMS::createFile . . . . .                         | 51        |
| 5.1.2    | Function FMS::createDir . . . . .                          | 52        |
| 5.1.3    | Function FMS::removeFile . . . . .                         | 52        |
| 5.1.4    | Function FMS::removeDir . . . . .                          | 53        |
| 5.1.5    | Function FMS::chGrp . . . . .                              | 54        |
| 5.1.6    | Function FMS::chMod . . . . .                              | 54        |
| 5.1.7    | Function FMS::headOfFile . . . . .                         | 55        |
| 5.1.8    | Function FMS::tailOfFile . . . . .                         | 56        |
| 5.1.9    | Function FMS::contentOfFile . . . . .                      | 57        |
| 5.1.10   | Function FMS::listDir . . . . .                            | 57        |
| 5.1.11   | Function FMS::copyFile . . . . .                           | 58        |
| 5.1.12   | Function FMS::copyAsyncFile . . . . .                      | 59        |
| 5.1.13   | Function FMS::moveFile . . . . .                           | 60        |
| 5.1.14   | Function FMS::moveAsyncFile . . . . .                      | 60        |
| 5.1.15   | Function FMS::stopFileTransfer . . . . .                   | 61        |
| 5.1.16   | Function FMS::listFileTransfers . . . . .                  | 62        |
| 5.1.17   | Function FMS::getFileInfo . . . . .                        | 63        |
| 5.2      | Data types definitions . . . . .                           | 63        |

# Chapter 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 Service
- TMS: Tasks Management Service
- FMS: Files Management Service
- IMS: Information Management Service

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 contains 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        | 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 be 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 Python 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.
- The VishnuInitialize() and VishnuFinalize() methods are not applicable to the WS API.
- Methods with restricted access (administration) are not 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 Python 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 parameters (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 Web Services description

The Web Services are fully described by the following documents which are attached to the current document:

- **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 References

- D1.1a : VISHNU General specifications

## 1.6 Glossary

None

## Chapter 2

# API specification for User Management Service (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      |
| session   | Session        | The session object that contains the created session details  | OUT  | yes      |
| options   | ConnectOptions | options is an object which encapsulates the options available for the connect method. It allows the user to choose the way for closing the session automatically on TIMEOUT or on DISCONNECT and the possibility for an admin to open a session as he/she was a specific user | IN   | no       |

##### 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       |
| ERRCODE_UNKNOWN_USER         | The user is unknown or the password is wrong |
| ERRCODE_UNKNOWN_CLOSURE_MODE | The closure policy is unknown                |
| ERRCODE_INCORRECT_TIMEOUT    | The value of the timeout is incorrect        |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                        |
| ERRCODE_NO_ADMIN             | The user is not an administrator             |
| ERRCODE_USER_LOCKED          | The user is locked                           |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)   |

| Name              | Description                                |
|-------------------|--|
| ERRCODE_DBERR     | Vishnu not available (Database error)      |
| ERRCODE_DBCONN    | Vishnu not available (Database connection) |
| ERRCODE_SYSTEM    | Vishnu not available (System)              |
| ERRCODE_UNDEFINED | Internal Error: Undefined exception        |

### Signature

int **vishnu::connect**(const string& userId, const string& password, Session& session, const ConnectOptions& options = ConnectOptions());

## 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      |
| session   | <b>Session</b> | The session object containing session information                  | OUT  | yes      |

### Description

The reconnect() function reconnects to a session that is still active

### 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            |
| ERRCODE_UNKNOWN_USER         | The user is unknown or the password is wrong      |
| ERRCODE_USER_LOCKED          | The user is locked                                |
| ERRCODE_UNUSABLE_MACHINE     | The machine does not exist or it is locked        |
| ERRCODE_UNKNOWN_SESSION_ID   | The session Id is unknown                         |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

int **vishnu::reconnect**(const string& userId, const string& password, const string& sessionId, Session& session);

## 2.1.3 Function UMS::addUser

### Access

This function can be used by ADMIN users only

**Parameters**



| Parameter  | Type   | Description   | Mode  | Required |
|------------|--------|---|-------|----------|
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU | IN    | yes      |
| newUser    | User   | Object containing the new user information                                    | INOUT | yes      |

### Description

The addUser() 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            |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_USER_LOCKED          | The user is locked                                |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_USERID_EXISTING      | The userId already exists in the database         |
| ERRCODE_INVALID_MAIL_ADRESS  | The mail address is invalid                       |
| ERRCODE_MACHINE_LOCKED       | The machine is locked                             |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

```
int vishnu::addUser(const string& sessionKey, 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 encrypted identifier of the session generated by VISHNU | IN   | yes      |
| user       | User   | Object containing user information  | IN   | yes      |

### Description

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

### Return Value

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

| Name                   | Description                            |
|------------------------|--|
| VISHNU_OK              | The service was performed successfully |
| ERRCODE_NO_ADMIN       | The user is not an administrator       |
| ERRCODE_UNKNOWN_USERID | The userId is unknown                  |

| Name                         | Description  |
|------------------------------|--|
| ERRCODE_INVALID_MAIL_ADRESS  | The mail address is invalid                          |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                      |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed.    |
| ERRCODE_USER_LOCKED          | The user is locked                                   |
| ERRCODE_USER_ALREADY_LOCKED  | Trying to lock a user account that is already locked |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal            |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)           |
| ERRCODE_DBERR                | Vishnu not available (Database error)                |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)           |
| ERRCODE_SYSTEM               | Vishnu not available (System)                        |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception                  |

### Signature

```
int vishnu::updateUser(const string& sessionKey, const User& user);
```

### 2.1.5 Function UMS::deleteUser

#### Access

This function can be used by ADMIN users only

#### Parameters

| Parameter  | Type   | Description  | Mode | Required |
|------------|--------|--|------|----------|
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU            | IN   | yes      |
| userId     | string | userId represents the VISHNU user identifier of the user who will be deleted from 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            |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_USER_LOCKED          | The user is locked                                |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

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



## 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 encrypted identifier of the session generated by VISHNU | IN   | yes      |

### Description

The close() function closes the 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            |
| ERRCODE_COMMAND_RUNNING      | Commands are running                              |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

```
int vishnu::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       |
| ERRCODE_UNKNOWN_USER | The user is unknown or the password is wrong |

| Name                | Description                                |
|---------------------|--|
| ERRCODE_USER_LOCKED | The user is locked                         |
| ERRCODE_DIET        | Vishnu not available (Service bus failure) |
| ERRCODE_DBERR       | Vishnu not available (Database error)      |
| ERRCODE_DBCONN      | Vishnu not available (Database connection) |
| ERRCODE_SYSTEM      | Vishnu not available (System)              |
| ERRCODE_UNDEFINED   | Internal Error: Undefined exception        |

### Signature

```
int vishnu::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 |
|-------------|--------|---|------|----------|
| sessionKey  | string | The sessionKey is the encrypted identifier of the session generated by VISHNU         | IN   | yes      |
| userId      | string | userId represents the VISHNU user identifier of the user whose password will be reset | IN   | yes      |
| tmpPassword | string | The temporary password generated by VISHNU  | OUT  | 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            |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_USER_LOCKED          | The user is locked                                |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

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

## 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 encrypted identifier of the session generated by VISHNU | IN   | yes      |
| newAccount   | LocalAccount | newAccount is the object which encapsulates the new local user configuration  | IN   | yes      |
| sshPublicKey | string       | The SSH public key generated by VISHNU for accessing a local account          | OUT  | 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                          |
| ERRCODE_LOCAL_ACCOUNT_EXIST  | The local account already exists                                |
| ERRCODE_MACHINE_LOCKED       | The machine is locked   |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown   |
| ERRCODE_UNKNOWN_MACHINE      | The machine id is unknown                                       |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                                 |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed.               |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal                       |
| ERRCODE_LOGIN_ALREADY_USED   | The system account login is already used by another vishnu user |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)                      |
| ERRCODE_DBERR                | Vishnu not available (Database error)                           |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)                      |
| ERRCODE_SYSTEM               | Vishnu not available (System)                                   |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception                             |
| ERRCODE_UNUSABLE_MACHINE     | The machine does not exist or it is locked                      |

#### Signature

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

### 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 encrypted 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   |
|-------------------------------|---|
| ERRCODE_UNKNOWN_USERID        | The userId is unknown   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                       |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                 |
| ERRCODE_SESSIONKEY_EXPIRED    | The sessionKey is expired. The session is closed.               |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local is unknown  |
| CLI_ERROR_NO_SESSION          | There is no open session in this terminal                       |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                      |
| ERRCODE_DBERR                 | Vishnu not available (Database error)                           |
| ERRCODE_DBCONN                | Vishnu not available (Database connection)                      |
| ERRCODE_SYSTEM                | Vishnu not available (System)                                   |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                             |
| ERRCODE_LOGIN_ALREADY_USED    | The system account login is already used by another vishnu user |

### Signature

```
int vishnu::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 |
|------------|--------|--|------|----------|
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU  | IN   | yes      |
| 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      |

### 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            |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local is unknown                              |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED    | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION          | There is no open session in this terminal         |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                 | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                | Vishnu not available (System)                     |

| Name              | Description                         |
|-------------------|-------------------------------------|
| ERRCODE_UNDEFINED | Internal Error: Undefined exception |

**Signature**

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

**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 encrypted 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           |
| ERRCODE_NO_ADMIN          | The user is not an administrator                 |
| ERRCODE_SAVE_CONFIG_ERROR | A problem occurs during the configuration saving |
| CLI_ERROR_NO_SESSION      | There is no open session in this terminal        |
| ERRCODE_DIET              | Vishnu not available (Service bus failure)       |
| ERRCODE_DBERR             | Vishnu not available (Database error)            |
| ERRCODE_DBCONN            | Vishnu not available (Database connection)       |
| ERRCODE_SYSTEM            | Vishnu not available (System)                    |
| ERRCODE_UNDEFINED         | Internal Error: Undefined exception              |

**Signature**

```
int vishnu::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 encrypted identifier of the session generated by VISHNU | IN   | yes      |
| filePath   | string | The filePath is the path of the file used to restore VISHNU configuration     | 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              |
| ERRCODE_NO_ADMIN             | The user is not an administrator                    |
| ERRCODE_RESTORE_CONFIG_ERROR | A problem occurs during the configuration restoring |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal           |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)          |
| ERRCODE_DBERR                | Vishnu not available (Database error)               |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)          |
| ERRCODE_SYSTEM               | Vishnu not available (System)                       |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception                 |

**Signature**

```
int vishnu::restoreConfiguration(const string& sessionKey, const string& filePath);
```

**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 encrypted identifier of the session generated by VISHNU  | IN    | yes      |
| newMachine | Machine | is an object which encapsulates the information of the machine which will be added in VISHNU except the machine id which will be created automatically by VISHNU | INOUT | 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            |
| ERRCODE_MACHINE_EXISTING     | The machineId already exists in the database      |
| ERRCODE_UNKNOWN_CLOSURE_MODE | The closure policy is unknown                     |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |



**Signature**

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

**2.1.15 Function UMS::updateMachine****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type    | Description   | Mode | Required |
|------------|---------|---|------|----------|
| sessionKey | string  | The sessionKey is the encrypted identifier of the session generated by VISHNU | IN   | yes      |
| machine    | Machine | existing machine information  | IN   | yes      |

**Description**

The updateMachine() function updates machine description

**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            |
| ERRCODE_UNKNOWN_CLOSURE_MODE | The closure policy is unknown                     |
| ERRCODE_UNKNOWN_MACHINE      | The machine id is unknown                         |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

**Signature**

```
int vishnu::updateMachine(const string& sessionKey, const Machine& machine);
```

**2.1.16 Function UMS::deleteMachine****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description   | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU | IN   | yes      |
| machineId  | string | machineId represents the identifier of the machine                            | 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            |
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_UNKNOWN_MACHINE      | The machine id is unknown                         |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

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

## 2.1.17 Function UMS::listLocalAccounts

### Access

This function can be used by any VISHNU user

### Parameters

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

### Description

The listLocalAccounts() 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            |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |



**Signature**

```
int vishnu::listLocalAccounts(const string& sessionKey, ListLocalAccounts& listLocalAcct, const ListLocalAccOptions& options = ListLocalAccOptions());
```

**2.1.18 Function UMS::listMachines****Access**

This function can be used by any VISHNU user

**Parameters**

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

**Description**

The listMachines() function lists the machines that are accessible through 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            |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

**Signature**

```
int vishnu::listMachines(const string& sessionKey, ListMachines& listMachine, const ListMachineOptions& options = ListMachineOptions());
```

**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 encrypted identifier of the session generated by VISHNU | IN   | yes      |
| listCommands | ListCommands | listCommands is the list of commands  | OUT  | yes      |

| Parameter | Type           | Description   | Mode | Required |
|-----------|----------------|---|------|----------|
| options   | ListCmdOptions | allows the user to list commands by using several optional criteria: a period, specific session and for admin to list all commands of all VISHNU users or commands from a specific user | IN   | no       |

### 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            |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

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

## 2.1.20 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      |
| listOptValues | ListOptionsValues | listOptValues is an object which encapsulates the list of options   | OUT  | yes      |
| options       | ListOptOptions    | allows to list a specific option or all default options values or for an admin to list options of a specific user | IN   | no       |

### 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 |
| ERRCODE_NO_ADMIN | The user is not an administrator       |

| Name                         | Description                                       |
|------------------------------|---|
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_UNKNOWN_OPTION       | The name of the user option is unknown            |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

```
int vishnu::listOptions(const string& sessionKey, ListOptionsValues& listOptValues, const ListOptOptions& options = ListOptOptions());
```

### 2.1.21 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      |
| listuser     | ListUsers | listuser is the list of users   | OUT  | yes      |
| userIdOption | string    | allows an admin to get information about a specific user identified by his/her userId | IN   | no       |

#### Description

The listUsers() function lists VISHNU users

#### Return Value

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

| Name                         | Description                                       |
|------------------------------|---|
| ERRCODE_NO_ADMIN             | The user is not an administrator                  |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

### Signature

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

### 2.1.22 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 encrypted identifier of the session generated by VISHNU  | IN   | yes      |
| listsession | ListSessions       | listsession is the list of sessions  | OUT  | 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       |

#### 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            |
| ERRCODE_UNKNOWN_USERID       | The userId is unknown                             |
| ERRCODE_UNKNOWN_CLOSURE_MODE | The closure policy is unknown                     |
| ERRCODE_SESSIONKEY_EXPIRED   | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND | The session key is unrecognized                   |
| CLI_ERROR_NO_SESSION         | There is no open session in this terminal         |
| ERRCODE_DIET                 | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                | Vishnu not available (Database error)             |
| ERRCODE_DBCONN               | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM               | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED            | Internal Error: Undefined exception               |

#### Signature

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

### 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 encrypted 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            |
| ERRCODE_NO_ADMIN               | The user is not an administrator                  |
| ERRCODE_UNKNOWN_OPTION         | The name of the user option is unknown            |
| ERRCODE_SESSIONKEY_EXPIRED     | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND   | The session key is unrecognized                   |
| ERRCODE_INCORRECT_TIMEOUT      | The value of the timeout is incorrect             |
| ERRCODE_INCORRECT_TRANSFER_CMD | The value of the transfer command is incorrect    |
| CLI_ERROR_NO_SESSION           | There is no open session in this terminal         |
| ERRCODE_DIET                   | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                  | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                 | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                 | Vishnu not available (System)                     |
| ERRCODE_UNDEFINED              | Internal Error: Undefined exception               |

**Signature**

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

**2.1.24 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 encrypted 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 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            |
| ERRCODE_UNKNOWN_OPTION         | The name of the user option is unknown            |
| ERRCODE_UNKNOWN_CLOSURE_MODE   | The closure policy is unknown                     |
| ERRCODE_SESSIONKEY_EXPIRED     | The sessionKey is expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND   | The session key is unrecognized                   |
| ERRCODE_INCORRECT_TIMEOUT      | The value of the timeout is incorrect             |
| ERRCODE_INCORRECT_TRANSFER_CMD | The value of the transfer command is incorrect    |
| CLI_ERROR_NO_SESSION           | There is no open session in this terminal         |
| ERRCODE_DIET                   | Vishnu not available (Service bus failure)        |

| Name              | Description                                |
|-------------------|--|
| ERRCODE_DBERR     | Vishnu not available (Database error)      |
| ERRCODE_DBCONN    | Vishnu not available (Database connection) |
| ERRCODE_SYSTEM    | Vishnu not available (System)              |
| ERRCODE_UNDEFINED | Internal Error: Undefined exception        |

**Signature**

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

**2.1.25 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     |
| ERRCODE_DIET      | Vishnu not available (Service bus failure) |
| ERRCODE_UNDEFINED | Internal Error: Undefined exception        |

**Signature**

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

**2.1.26 Function UMS::vishnuFinalize****Access**

This function can be used by any VISHNU user

**Parameters**

This command defines no parameters.

**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     |
| ERRCODE_DIET      | Vishnu not available (Service bus failure) |
| ERRCODE_UNDEFINED | Internal Error: Undefined exception        |



**Signature**

```
int vishnu::vishnuFinalize();
```

**2.2 Data types definitions****Class UMS::Command 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)         |
| status         | CommandStatusType | The machineId is the identifier of the machine used by the command              |

**Class UMS::Configuration Content**

| Name                  | Type                 | Description   |
|-----------------------|----------------------|---|
| listConfUsers         | List of User         | is the list of users objects                                    |
| listConfMachines      | List of Machine      | is a list of machines objects                                   |
| listConfLocalAccounts | List of LocalAccount | is the list of LocalAccount objects                             |
| filePath              | string               | the full path to the file that contains the saved configuration |

**Class UMS::ConnectOptions Content**

| Name                   | Type             | Description  |
|------------------------|------------------|--|
| closePolicy            | SessionCloseType | is an option for closing session automatically   |
| sessionInactivityDelay | int              | is the maximum delay in seconds between two user requests when the CLOSE_ON_TIMEOUT policy is set              |
| substituteUserId       | string           | is an admin option which allows an admin to open a session as if she was another user identified by her userId |

**Class UMS::ListCmdOptions 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 |

**Class UMS::ListCommands Content**

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

**Class UMS::ListLocalAccOptions 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   |

**Class UMS::ListLocalAccounts Content**

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

**Class UMS::ListMachineOptions Content**

| Name           | Type    | Description  |
|----------------|---------|--|
| userId         | string  | is an admin option for listing machines in which a specific user has a local configuration |
| listAllmachine | boolean | is an option for listing all VISHNU machines   |
| machineId      | string  | is an option for listing information about a specific machine                              |

**Class UMS::ListMachines Content**

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

**Class UMS::ListOptOptions Content**

| Name             | Type    | Description  |
|------------------|---------|--|
| listAllDeftValue | boolean | is an option for listing all default option values defined by VISHNU administrator |
| userId           | string  | is an admin option for listing the options of a specific user                      |
| optionName       | string  | allows the user to get the value of a specific option identified by its name       |

**Class UMS::ListOptionsValues Content**

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



**Class UMS::ListSessionOptions Content**

| Name                   | Type             | Description  |
|------------------------|------------------|--|
| status                 | StatusType       | specifies the status of the sessions which will be listed  |
| sessionClosePolicy     | SessionCloseType | specifies the closure mode of the sessions which will be listed (CLOSE_ON_TIMEOUT or CLOSE_ON_DISCONNECT)  |
| sessionInactivityDelay | int              | specifies the inactivity delay in seconds 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 |

**Class UMS::ListSessions Content**

| Name     | Type            | Description                    |
|----------|-----------------|--------------------------------|
| sessions | List of Session | is the list of session objects |

**Class UMS::ListUsers Content**

| Name  | Type         | Description                  |
|-------|--------------|------------------------------|
| users | List of User | is the list of users objects |

**Class UMS::LocalAccount 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             |

**Class UMS::Machine 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    |
| machineDescription | string | represents the description of the machine |

| Name         | Type       | Description  |
|--------------|------------|--|
| language     | string     | represents the language used for the description of the machine          |
| status       | StatusType | represents the status of the machine                                     |
| sshPublicKey | string     | contains the SSH public key used by VISHNU to access local user accounts |

**Class UMS::OptionValue Content**

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

**Class UMS::Session 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)          |
| status          | StatusType       | represents the status of the session   |
| closePolicy     | SessionCloseType | is the way to close the session  |
| timeout         | long             | is the inactivity delay in seconds associated to the CLOSE_ON_TIMEOUT option     |

**Class UMS::User 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 | PrivilegeType | is the privilege of the user (admin or simple user)   |
| email     | string        | is the email of the user  |
| status    | StatusType    | represents the status of the user   |

**Enumeration UMS::CommandStatusType Type**

| Name    | Value |
|---------|-------|
| FAILURE | 0     |
| SUCCESS | 1     |

**Enumeration UMS::PrivilegeType Type**

| Name | Value |
|------|-------|
| USER | 0     |

| Name  | Value |
|-------|-------|
| ADMIN | 1     |

**Enumeration UMS::SessionCloseType Type**

| Name                | Value |
|---------------------|-------|
| UNDEFINED           | 0     |
| CLOSE_ON_TIMEOUT    | 1     |
| CLOSE_ON_DISCONNECT | 2     |

**Enumeration UMS::StatusType Type**

| Name     | Value |
|----------|-------|
| INACTIVE | 0     |
| ACTIVE   | 1     |

## Chapter 3

# API specification for Tasks Management Service (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 session key   | IN   | yes      |
| machineId      | string        | Is the id of the machine on which 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      |
| jobInfo        | Job           | The Job object containing the output information (ex: jobId and jobPath) of the job to submit   | 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 a job on a machine through the use of a script (scriptFilePath)

##### 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            |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_INVALID_PARAM           | Error invalid parameters                          |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |
| ERRCODE_SSH                     | Vishnu not available (SSH error)                  |

### Signature

int **vishnu::submitJob**(const string& sessionKey, const string& machineId, const string& scriptFilePath, Job& jobInfo, const SubmitOptions& options = SubmitOptions());

### 3.1.2 Function TMS::getJobInfo

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter  | Type       | Description  | Mode | Required |
|------------|------------|--|------|----------|
| sessionKey | string     | The session key                                      | IN   | yes      |
| machineId  | string     | Is the id of the machine on which the job is running | IN   | yes      |
| jobId      | string     | The id of the job                                    | IN   | yes      |
| jobInfos   | <b>Job</b> | 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                                       |
|---------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| VISHNU_OK                       | The service was performed successfully            |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |

### Signature

int **vishnu::getJobInfo**(const string& sessionKey, const string& machineId, const string& jobId, Job& jobInfos);

### 3.1.3 Function TMS::getJobProgress

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter    | Type            | Description  | Mode | Required |
|--------------|-----------------|--|------|----------|
| sessionKey   | string          | The session key  | IN   | yes      |
| machineId    | string          | Is the id of the machine to get the jobs progression                 | IN   | yes      |
| listProgress | ListProgression | 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 gets 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            |
| ERRCODE_UNKNOWN_MACHINE    | The machine id is unknown                         |
| ERRCODE_SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| ERRCODE_UNDEFINED          | Internal Error: Undefined exception               |
| ERRCODE_DIET               | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR              | Vishnu not available (Database error)             |
| ERRCODE_DBCONN             | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM             | Vishnu not available (System)                     |

#### Signature

```
int vishnu::getJobProgress(const string& sessionKey, const string& machineId, ListProgression& listProgress, const ProgressOptions& options = ProgressOptions());
```

### 3.1.4 Function TMS::listQueues

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter    | Type       | Description   | Mode | Required |
|--------------|------------|---|------|----------|
| sessionKey   | string     | The session key   | IN   | yes      |
| machineId    | string     | Is the id of the machine that the user wants to list queues     | IN   | yes      |
| listofQueues | ListQueues | The list of queues  | OUT  | yes      |
| queueName    | string     | if it is given, listQueues gives information only of this queue | IN   | no       |

#### 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                                       |
|---------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| VISHNU_OK                       | The service was performed successfully            |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |

### Signature

int **vishnu::listQueues**(const string& sessionKey, const string& machineId, ListQueues& listofQueues, const string& queueName = string());

### 3.1.5 Function TMS::listJobs

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter  | Type            | Description  | Mode | Required |
|------------|-----------------|--|------|----------|
| sessionKey | string          | The session key  | IN   | yes      |
| machineId  | string          | Is the id of the machine on which the jobs are running | 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                                       |
|---------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| VISHNU_OK                       | The service was performed successfully            |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |

### Signature

int **vishnu::listJobs**(const string& sessionKey, const string& machineId, ListJobs& listOfJobs, const ListJobsOptions& options = ListJobsOptions());



### 3.1.6 Function TMS::getJobOutput

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter  | Type             | Description  | Mode | Required |
|------------|------------------|--|------|----------|
| sessionKey | string           | The session key  | IN   | yes      |
| machineId  | string           | gets outputPath and errorPath of a job from its id   | IN   | yes      |
| jobId      | string           | The Id of the job  | IN   | yes      |
| outputInfo | <b>JobResult</b> | The Job object containing the job output information (ex: outputPath and errorPath) of the job to submit | OUT  | yes      |
| outDir     | string           | The output directory where the files will be stored (default is current directory)                       | IN   | no       |

#### Description

The getJobOutput() function gets standard output and error output files of a job given its id

#### Return Value

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

| Name                            | Description                                       |
|---------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| VISHNU_OK                       | The service was performed successfully            |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |
| ERRCODE_JOB_IS_NOT_TERMINATED   | The job is not terminated                         |
| ERRCODE_SSH                     | Vishnu not available (SSH error)                  |
| ERRCODE_ALREADY_DOWNLOADED      | The job is already downloaded                     |

#### Signature

```
int vishnu::getJobOutput(const string& sessionKey, const string& machineId, const string& jobId, JobResult& outputInfo, const string& outDir = string());
```

### 3.1.7 Function TMS::getCompletedJobsOutput

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter     | Type                  | Description   | Mode | Required |
|---------------|-----------------------|---|------|----------|
| sessionKey    | string                | The session key   | IN   | yes      |
| machineId     | string                | Is the id of the machine on which the jobs are been submitted | IN   | yes      |
| listOfResults | <b>ListJobResults</b> | Is the list of jobs results                                   | OUT  | yes      |



| Parameter | Type   | Description  | Mode | Required |
|-----------|--------|--|------|----------|
| outDir    | string | Specifies the output directory where the files will be stored (by default, the current directory). | IN   | no       |

### Description

The getCompletedJobsOutput() function gets standard output and error output files of completed jobs (applies only once for each job)

### Return Value

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

| Name                            | Description                                       |
|---------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR   | The batch scheduler indicates an error            |
| ERRCODE_SESSIONKEY_EXPIRED      | The sessionKey is expired. The session is closed. |
| VISHNU_OK                       | The service was performed successfully            |
| ERRCODE_PERMISSION_DENIED       | Permission denied                                 |
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown               |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                         |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception               |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure)        |
| ERRCODE_DBERR                   | Vishnu not available (Database error)             |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection)        |
| ERRCODE_SYSTEM                  | Vishnu not available (System)                     |
| ERRCODE_SSH                     | Vishnu not available (SSH error)                  |

### Signature

```
int vishnu::getCompletedJobsOutput(const string& sessionKey, const string& machineId, ListJobResults& listOfResults, const string& outDir = string());
```

## 3.1.8 Function TMS::cancelJob

### Access

This function can be used by any VISHNU user

### Parameters

| Parameter  | Type   | Description  | Mode | Required |
|------------|--------|--|------|----------|
| sessionKey | string | The session key                                      | IN   | yes      |
| machineId  | string | Is the id of the machine on which the job is running | IN   | yes      |
| jobId      | string | The Id of the job                                    | IN   | yes      |

### Description

The cancelJob() function cancels a job from its id

### Return Value

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

| Name                          | Description                                       |
|-------------------------------|---|
| ERRCODE_BATCH_SCHEDULER_ERROR | The batch scheduler indicates an error            |
| ERRCODE_PERMISSION_DENIED     | Permission denied                                 |
| VISHNU_OK                     | The service was performed successfully            |
| ERRCODE_SESSIONKEY_EXPIRED    | The sessionKey is expired. The session is closed. |

| Name                            | Description                                |
|---------------------------------|--|
| ERRCODE_UNKNOWN_BATCH_SCHEDULER | The batch scheduler type is unknown        |
| ERRCODE_UNKNOWN_MACHINE         | The machine id is unknown                  |
| ERRCODE_UNDEFINED               | Internal Error: Undefined exception        |
| ERRCODE_DIET                    | Vishnu not available (Service bus failure) |
| ERRCODE_DBERR                   | Vishnu not available (Database error)      |
| ERRCODE_DBCONN                  | Vishnu not available (Database connection) |
| ERRCODE_SYSTEM                  | Vishnu not available (System)              |
| ERRCODE_ALREADY_CANCELED        | The job is already canceled                |
| ERRCODE_ALREADY_TERMINATED      | The job is already terminated              |
| ERRCODE_SSH                     | Vishnu not available (SSH error)           |

### Signature

int **vishnu::cancelJob**(const string& sessionKey, const string& machineId, const string& jobId);

## 3.2 Data types definitions

### Class TMS::Job 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 | string      | Is the number of nodes and processors per node used by the job.            |

### Class TMS::JobResult Content

| Name  | Type   | Description                   |
|-------|--------|-------------------------------|
| jobId | string | Represents the id of the job. |

| Name       | Type   | Description  |
|------------|--------|--|
| 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. |

**Class TMS::ListJobResults Content**

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

**Class TMS::ListJobs 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 <b>Job</b> | Is a list of job information (jobId, jobName, ...).      |

**Class TMS::ListJobsOptions Content**

| Name           | Type               | Description  |
|----------------|--------------------|--|
| jobId          | string             | lists the job with the specified id                                |
| nbCpu          | int                | lists the jobs with the specified number of CPUs                   |
| fromSubmitDate | long               | lists the jobs submitted after the specified date (UNIX timestamp) |
| toSubmitDate   | long               | lists jobs submitted before the specified date (UNIX timestamp)    |
| owner          | string             | lists the jobs submitted by the specified owner                    |
| status         | <b>JobStatus</b>   | lists the jobs with the specified status                           |
| priority       | <b>JobPriority</b> | lists the jobs with the specified priority                         |
| queue          | string             | the jobs with the specified queue name                             |

**Class TMS::ListProgression Content**

| Name     | Type                       | Description  |
|----------|----------------------------|--|
| nbJobs   | int                        | Represents the number of jobs in progression list. |
| progress | List of <b>Progression</b> | Represents the list of jobs in progression.        |

**Class TMS::ListQueues Content**

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

**Class TMS::ProgressOptions Content**

| Name     | Type   | Description  |
|----------|--------|--|
| jobId    | string | Specifies the id of the job whose progression the user wants to see. |
| jobOwner | string | Specifies the owner of the job.                                      |

**Class TMS::Progression 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   | int       | Represent the job progression.                  |
| status    | JobStatus | Represents the job status.                      |

**Class TMS::Queue 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         | QueueStatus   | Is the status of the queue.                       |
| priority      | QueuePriority | Represents the priority of the queue.             |
| description   | string        | Is the queue description.                         |

**Class TMS::SubmitOptions Content**

| Name                 | Type   | Description  |
|----------------------|--------|--|
| name                 | string | Assigns a job name. The default is the path of 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    | Is the memory size that the job requires                 |
| nbCpu                | int    | The number of cpu that the job requires                  |
| nbNodesAndCpuPerNode | string | 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                  |

**Enumeration TMS::JobPriority Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | -1    |
| VERY_LOW  | 1     |
| LOW       | 2     |
| NORMAL    | 3     |
| HIGH      | 4     |
| VERY_HIGH | 5     |

**Enumeration TMS::JobStatus Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | -1    |
| SUBMITTED | 1     |

| Name               | Value |
|--------------------|-------|
| QUEUED             | 2     |
| WAITING            | 3     |
| RUNNING            | 4     |
| TERMINATED         | 5     |
| CANCELLED          | 6     |
| ALREADY_DOWNLOADED | 7     |

**Enumeration TMS::QueuePriority Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | -1    |
| VERY_LOW  | 1     |
| LOW       | 2     |
| NORMAL    | 3     |
| HIGH      | 4     |
| VERY_HIGH | 5     |

**Enumeration TMS::QueueStatus Type**

| Name        | Value |
|-------------|-------|
| NOT_STARTED | 0     |
| STARTED     | 1     |
| RUNNING     | 2     |

## Chapter 4

# API specification for Information Management Service (IMS)

### 4.1 Definition of the functions of the package

#### 4.1.1 Function IMS::exportCommands

##### Access

This function can be used by any VISHNU user

##### Parameters

| Parameter    | Type     | Description  | Mode  | Required |
|--------------|----------|--|-------|----------|
| sessionKey   | string   | The session key  | IN    | yes      |
| oldSessionId | string   | The id of the session to export (session has ended)              | IN    | yes      |
| filename     | string   | The path of the output file containing the Vishnu shell commands | INOUT | yes      |
| options      | ExportOp | options which encapsulate the option for the export              | IN    | no       |

##### Description

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

##### Return Value

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

| Name                          | Description   |
|-------------------------------|---|
| VISHNU_OK                     | Error code returned if success                      |
| ERRCODE_DB_ERROR              | The database generated an error                     |
| ERRCODE_INVALID_PARAMETER     | If a parameter is invalid                           |
| ERRCODE_UNDEFINED             | Undefined error code                                |
| ERRCODE_SESSION_KEY_EXPIRED   | The session key has expired. The session is closed. |
| ERRCODE_SESSION_KEY_NOT_FOUND | The session key is unrecognized.                    |

##### Signature

```
int vishnu::exportCommands(const string& sessionKey, const string& oldSessionId, string& filename, const ExportOp& options = ExportOp());
```

### 4.1.2 Function IMS::getMetricCurrentValue

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter   | Type        | Description                              | Mode | Required |
|-------------|-------------|--|------|----------|
| sessionKey  | string      | The session key                          | IN   | yes      |
| machineId   | string      | The id of the machine                    | IN   | yes      |
| metricValue | ListMetric  | Value of the metric                      | OUT  | yes      |
| options     | CurMetricOp | The options for the current metric value | IN   | yes      |

#### Description

The getMetricCurrentValue() function displays the current values of system metrics

#### 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                      |
| ERRCODE_DB_ERROR              | The database generated an error                     |
| ERRCODE_INVALID_PARAMETER     | If a parameter is invalid                           |
| ERRCODE_SESSION_KEY_EXPIRED   | The session key has expired. The session is closed. |
| ERRCODE_SESSION_KEY_NOT_FOUND | The session key is unrecognized.                    |
| ERRCODE_UNDEFINED             | Undefined error code                                |

#### Signature

```
int vishnu::getMetricCurrentValue(const string& sessionKey, const string& machineId, ListMetric& metricValue, const CurMetricOp& options);
```

### 4.1.3 Function IMS::getMetricHistory

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter    | Type         | Description                                | Mode | Required |
|--------------|--------------|--|------|----------|
| sessionKey   | string       | The session key                            | IN   | yes      |
| machineId    | string       | The id of the machine                      | IN   | yes      |
| metricValues | ListMetric   | List of metric values                      | OUT  | yes      |
| options      | MetricHistOp | The optional fields for the metric history | IN   | no       |

#### Description

The getMetricHistory() function displays the history of values of a system 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  |
| ERRCODE_DB_ERROR | The database generated an error |



| Name                          | Description   |
|-------------------------------|---|
| ERRCODE_INVALID_PARAMETER     | If a parameter is invalid                           |
| ERRCODE_SESSION_KEY_EXPIRED   | The session key has expired. The session is closed. |
| ERRCODE_SESSION_KEY_NOT_FOUND | The session key is unrecognized.                    |
| ERRCODE_UNDEFINED             | Undefined error code                                |

**Signature**

```
int vishnu::getMetricHistory(const string& sessionKey, const string& machineId, ListMetric& metricValues, const MetricHistOp& options = MetricHistOp());
```

**4.1.4 Function IMS::getProcesses****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type          | Description                                     | Mode | Required |
|------------|---------------|---|------|----------|
| sessionKey | string        | The session key                                 | IN   | yes      |
| process    | ListProcesses | The list of the Vishnu processes on the machine | OUT  | yes      |
| options    | ProcessOp     | The options to search for the processes         | IN   | no       |

**Description**

The getProcesses() function displays the list of the VISHNU processes running on 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 |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid      |

**Signature**

```
int vishnu::getProcesses(const string& sessionKey, ListProcesses& process, const ProcessOp& options = ProcessOp());
```

**4.1.5 Function IMS::setSystemInfo****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type       | Description   | Mode | Required |
|------------|------------|---|------|----------|
| sessionKey | string     | The session key   | IN   | yes      |
| systemInfo | SystemInfo | Contains system information to store in Vishnu database | IN   | yes      |

**Description**

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

**Return Value**

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



| Name                      | Description                     |
|---------------------------|---------------------------------|
| VISHNU_OK                 | Error code returned if success  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

```
int vishnu::setSystemInfo(const string& sessionKey, const SystemInfo& systemInfo);
```

**4.1.6 Function IMS::setSystemThreshold****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type      | Description          | Mode | Required |
|------------|-----------|----------------------|------|----------|
| sessionKey | string    | The session key      | IN   | yes      |
| threshold  | Threshold | The threshold to set | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

```
int vishnu::setSystemThreshold(const string& sessionKey, const Threshold& threshold);
```

**4.1.7 Function IMS::getSystemThreshold****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type          | Description                   | Mode | Required |
|------------|---------------|-------------------------------|------|----------|
| sessionKey | string        | The session key               | IN   | yes      |
| value      | ListThreshold | The thresholds value          | OUT  | yes      |
| options    | ThresholdOp   | The options for the threshold | IN   | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

```
int vishnu::getSystemThreshold(const string& sessionKey, ListThreshold& value, const ThresholdOp& options);
```

**4.1.8 Function IMS::defineUserIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description           | Mode | Required |
|------------|--------|-----------------------|------|----------|
| sessionKey | string | The session key       | 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  |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |
| ERRCODE_DB_ERROR          | The database generated an error |

**Signature**

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

**4.1.9 Function IMS::defineMachineIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description           | Mode | Required |
|------------|--------|-----------------------|------|----------|
| sessionKey | string | The session key       | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

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

**4.1.10 Function IMS::defineJobIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description           | Mode | Required |
|------------|--------|-----------------------|------|----------|
| sessionKey | string | The session key       | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

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

**4.1.11 Function IMS::defineTransferIdentifier****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description           | Mode | Required |
|------------|--------|-----------------------|------|----------|
| sessionKey | string | The session key       | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

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

**4.1.12 Function IMS::loadShed****Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

The loadShed() function sheds load 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  |
| ERRCODE_COMPONENT_ERROR   | If a component is unavailable   |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

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

**4.1.13 Function IMS::setUpdateFrequency****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type   | Description                               | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key                           | 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  |
| ERRCODE_DB_ERROR          | The database generated an error |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

**Signature**

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

**4.1.14 Function IMS::getUpdateFrequency****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type   | Description                               | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key                           | 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                      |
| ERRCODE_DB_ERROR              | The database generated an error                     |
| ERRCODE_INVALID_PARAMETER     | If a parameter is invalid                           |
| ERRCODE_SESSION_KEY_EXPIRED   | The session key has expired. The session is closed. |
| ERRCODE_SESSION_KEY_NOT_FOUND | The session key is unrecognized.                    |
| ERRCODE_UNDEFINED             | Undefined error code                                |

**Signature**

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

**4.1.15 Function IMS::stop****Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter  | Type    | Description   | Mode | Required |
|------------|---------|---|------|----------|
| sessionKey | string  | The session key                                       | IN   | yes      |
| process    | Process | The process to stop and do not try to restart anymore | IN   | yes      |

**Description**

The stop() function to stop (and do not try to relaunch) a SeD

### Return Value

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

| Name                      | Description                     |
|---------------------------|---------------------------------|
| ERRCODE_DB_ERROR          | The database generated an error |
| VISHNU_OK                 | Error code returned if success  |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |

### Signature

```
int vishnu::stop(const string& sessionKey, const Process& process);
```

## 4.1.16 Function IMS::getSystemInfo

### Access

This function can be used by any VISHNU user

### Parameters

| Parameter  | Type        | Description                               | Mode | Required |
|------------|-------------|---|------|----------|
| sessionKey | string      | The session key                           | IN   | yes      |
| res        | ListSysInfo | The list of the system information gotten | OUT  | yes      |
| options    | SysInfoOp   | Optional field for system information     | IN   | no       |

### Description

The getSystemInfo() function to get the system info on a machine

### Return Value

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

| Name                          | Description   |
|-------------------------------|---|
| ERRCODE_DB_ERROR              | The database generated an error                     |
| ERRCODE_INVALID_PARAMETER     | If a parameter is invalid                           |
| VISHNU_OK                     | Error code returned if success                      |
| ERRCODE_SESSION_KEY_EXPIRED   | The session key has expired. The session is closed. |
| ERRCODE_SESSION_KEY_NOT_FOUND | The session key is unrecognized.                    |
| ERRCODE_UNDEFINED             | Undefined error code                                |

### Signature

```
int vishnu::getSystemInfo(const string& sessionKey, ListSysInfo& res, const SysInfoOp& options = SysInfoOp());
```

## 4.1.17 Function IMS::restart

### Access

This function can be used by ADMIN users only

### Parameters

| Parameter  | Type      | Description                            | Mode | Required |
|------------|-----------|--|------|----------|
| sessionKey | string    | The session key                        | IN   | yes      |
| machineId  | string    | The id of the machine where to restart | IN   | yes      |
| options    | RestartOp | The option for the restart             | IN   | yes      |

**Description**

The restart() function to restart a SeD or a MA

**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  |
| ERRCODE_INVALID_PARAMETER | If a parameter is invalid       |
| ERRCODE_DB_ERROR          | The database generated an error |

**Signature**

```
int vishnu::restart(const string& sessionKey, const string& machineId, const RestartOp& options);
```

## 4.2 Data types definitions

**Class IMS::CurMetricOp Content**

| Name       | Type              | Description            |
|------------|-------------------|------------------------|
| metricType | <b>MetricType</b> | The type of the metric |

**Class IMS::ExportOp Content**

| Name       | Type              | Description        |
|------------|-------------------|--------------------|
| exportType | <b>ExportType</b> | The type to export |

**Class IMS::ListMetric Content**

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

**Class IMS::ListProcesses Content**

| Name    | Type                   | Description               |
|---------|------------------------|---------------------------|
| process | List of <b>Process</b> | The processes of the list |

**Class IMS::ListSysInfo Content**

| Name    | Type                      | Description            |
|---------|---------------------------|------------------------|
| sysInfo | List of <b>SystemInfo</b> | The set of system info |

**Class IMS::ListThreshold Content**

| Name      | Type                     | Description                |
|-----------|--------------------------|----------------------------|
| Threshold | List of <b>Threshold</b> | The list of the thresholds |

**Class IMS::Metric Content**

| Name | Type              | Description            |
|------|-------------------|------------------------|
| type | <b>MetricType</b> | The type of the metric |

| Name  | Type | Description                            |
|-------|------|--|
| value | long | The value of the metric                |
| time  | long | The timestamp the metric had the value |

**Class IMS::MetricHistOp Content**

| Name      | Type       | Description                       |
|-----------|------------|-----------------------------------|
| startTime | long       | The start time to get the history |
| endTime   | long       | The end time to get the history   |
| type      | MetricType | The type of metric searched       |

**Class IMS::Process Content**

| Name        | Type         | Description  |
|-------------|--------------|--|
| processName | string       | The name of the process (the name of the executable)                   |
| machineId   | string       | The id of the machine  |
| dietId      | string       | The diet id of the process   |
| state       | ProcessState | The state of the process   |
| timestamp   | long         | The timestamp corresponding to the moment the process has been updated |
| script      | string       | The content of the script used to launch the sed                       |

**Class IMS::ProcessOp Content**

| Name      | Type   | Description           |
|-----------|--------|-----------------------|
| machineId | string | The id of the machine |

**Class IMS::RestartOp Content**

| Name       | Type    | Description                               |
|------------|---------|---|
| vishnuConf | string  | The path to the vishnu configuration file |
| sedType    | SeDType | The type of the vishnu sed                |

**Class IMS::SysInfoOp Content**

| Name      | Type   | Description    |
|-----------|--------|----------------|
| machineId | string | The machine id |

**Class IMS::SystemInfo Content**

| Name      | Type   | Description  |
|-----------|--------|--|
| memory    | long   | Amount of RAM memory available on the machine (in Bytes) |
| diskSpace | long   | Amount of disk space available on the machine (in Bytes) |
| machineId | string | The id of the machine                                    |

**Class IMS::Threshold Content**

| Name      | Type   | Description                               |
|-----------|--------|---|
| value     | long   | The value of the threshold                |
| machineId | string | The machine ID the threshold is available |



| Name    | Type       | Description   |
|---------|------------|---|
| type    | MetricType | The type of the threshold   |
| handler | string     | The userId of the admin responsible for handling the threshold problems |

**Class IMS::ThresholdOp Content**

| Name       | Type       | Description                                       |
|------------|------------|---|
| machineId  | string     | The id of the machine where the metric is defined |
| metricType | MetricType | The type of the metric                            |

**Enumeration IMS::ExportType Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | 0     |
| SHELL     | 1     |

**Enumeration IMS::LoadShedType Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | 0     |
| HARD      | 1     |
| SOFT      | 2     |

**Enumeration IMS::MetricType Type**

| Name          | Value |
|---------------|-------|
| UNDEFINED     | 0     |
| CPUUSE        | 1     |
| FREEDISKSPACE | 2     |
| FREEMEMORY    | 3     |

**Enumeration IMS::ProcessState Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | 0     |
| RUNNING   | 1     |
| DOWN      | 2     |
| DELETED   | 3     |

**Enumeration IMS::SeDType Type**

| Name      | Value |
|-----------|-------|
| UNDEFINED | 0     |
| UMS       | 1     |
| TMS       | 2     |
| FMS       | 3     |
| IMS       | 4     |

## Chapter 5

# API specification for File Management Service (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 |
|------------|--------|--|------|----------|
| sessionKey | string | The session key.   | IN   | yes      |
| path       | string | The file to create following the pattern [host:]file path. | IN   | yes      |

##### 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

**Signature**

```
int vishnu::createFile(const string& sessionKey, const string& path);
```

**5.1.2 Function FMS::createDir****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type   | Description  | Mode | Required |
|------------|--------|--|------|----------|
| sessionKey | string | The session key  | IN   | yes      |
| path       | string | The directory to create following the pattern [host:]directory path. | IN   | yes      |

**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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

**Signature**

```
int vishnu::createDir(const string& sessionKey, const string& path);
```

**5.1.3 Function FMS::removeFile****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type          | Description   | Mode | Required |
|------------|---------------|---|------|----------|
| sessionKey | string        | The session key   | IN   | yes      |
| path       | string        | The file to remove following the pattern [host:]file path | IN   | yes      |
| options    | RmFileOptions | the remove command options                                | IN   | no       |

**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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

**Signature**

```
int vishnu::removeFile(const string& sessionKey, const string& path, const RmFileOptions& options = RmFileOptions());
```

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

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type   | Description   | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key   | IN   | yes      |
| path       | string | The directory to remove 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.                    |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |

| Name                          | Description   |
|-------------------------------|---|
| ERRCODE_RUNTIME_ERROR         | Runtime error                                       |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed. |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                     |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                 |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                        |
| VISHNU_OK                     | The service has been performed succesfully.         |

### Signature

```
int vishnu::removeDir(const string& sessionKey, const string& path);
```

## 5.1.5 Function FMS::chGrp

### Access

This function can be used by any VISHNU user

### Parameters

| Parameter  | Type   | Description   | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key   | IN   | yes      |
| group      | string | the new group owner of file/directory                     | IN   | yes      |
| path       | string | The file/directory following the pattern [host:]file path | IN   | yes      |

### Description

The chGrp() function changes group owner of remote files/directories.

### Return Value

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

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

### Signature

```
int vishnu::chGrp(const string& sessionKey, const string& group, const string& path);
```

## 5.1.6 Function FMS::chMod

### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter  | Type   | Description   | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key   | IN   | yes      |
| mode       | mode_t | the access rights of file/directory in octal system.      | IN   | yes      |
| path       | string | The file/directory following the pattern [host:]file path | IN   | yes      |

#### Description

The chMod() function changes access rights of remote files/directories.

#### Return Value

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

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id is unknown                                     |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed successfully.                   |

#### Signature

```
int vishnu::chMod(const string& sessionKey, const mode_t& mode, const string& path);
```

### 5.1.7 Function FMS::headOfFile

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter   | Type              | Description                                     | Mode | Required |
|-------------|-------------------|---|------|----------|
| sessionKey  | string            | The session key                                 | IN   | yes      |
| path        | string            | The file following the pattern [host:]file path | IN   | yes      |
| fileContent | string            | The first "nLine" lines of the file             | OUT  | yes      |
| options     | HeadOfFileOptions | the head command options                        | IN   | no       |

#### 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

### Signature

```
int vishnu::headOfFile(const string& sessionKey, const string& path, string& fileContent, const HeadOfFileOptions& options = HeadOfFileOptions());
```

### 5.1.8 Function FMS::tailOfFile

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter   | Type                     | Description                                     | Mode | Required |
|-------------|--------------------------|---|------|----------|
| sessionKey  | string                   | The session key                                 | IN   | yes      |
| path        | string                   | The file following the pattern [host:]file path | IN   | yes      |
| fileContent | string                   | The last "nLine" lines of the file              | OUT  | yes      |
| options     | <b>TailOfFileOptions</b> | the tail command options                        | IN   | no       |

#### 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |



**Signature**

```
int vishnu::tailOfFile(const string& sessionKey, const string& path, string& fileContent, const TailOfFileOptions& options = TailOfFileOptions());
```

**5.1.9 Function FMS::contentOfFile****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter   | Type   | Description  | Mode | Required |
|-------------|--------|--|------|----------|
| sessionKey  | string | The session key  | IN   | yes      |
| path        | string | The file to display following the pattern [host:]file path | IN   | yes      |
| fileContent | string | The content of the file                                    | OUT  | yes      |

**Description**

The contentOfFile() function displays content 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id is unknown                                     |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed successfully.                   |

**Signature**

```
int vishnu::contentOfFile(const string& sessionKey, const string& path, string& fileContent);
```

**5.1.10 Function FMS::listDir****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type   | Description   | Mode | Required |
|------------|--------|---|------|----------|
| sessionKey | string | The session key   | IN   | yes      |
| path       | string | The directory to list following the pattern [host:]directory path | IN   | yes      |

| Parameter  | Type         | Description   | Mode | Required |
|------------|--------------|---|------|----------|
| dirContent | StringList   | The content of the directory (only files names with short format, all files informations with long format). | OUT  | yes      |
| options    | LsDirOptions | List of options for the listDir command   | IN   | yes      |

### Description

The listDir() function displays the content of a remote directory.

### Return Value

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

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

### Signature

```
int vishnu::listDir(const string& sessionKey, const string& path, StringList& dirContent, const LsDirOptions& options);
```

## 5.1.11 Function FMS::copyFile

### Access

This function can be used by any VISHNU user

### Parameters

| Parameter  | Type          | Description  | Mode | Required |
|------------|---------------|--|------|----------|
| sessionKey | string        | The session key  | IN   | yes      |
| src        | string        | The source file to copy following the pattern [host:]file path | IN   | yes      |
| dest       | string        | The path of the destination file                               | IN   | yes      |
| options    | CpFileOptions | 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                       |
|------------------------|-----------------------------------|
| ERRCODE_CONFIGNOTFOUND | Undefined configuration parameter |

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |

### Signature

```
int vishnu::copyFile(const string& sessionKey, const string& src, const string& dest, const CpFileOptions& options = CpFileOptions());
```

### 5.1.12 Function FMS::copyAsyncFile

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter    | Type          | Description   | Mode | Required |
|--------------|---------------|---|------|----------|
| sessionKey   | string        | The session key   | IN   | yes      |
| src          | string        | The source file to copy following the pattern [host:]file path  | IN   | yes      |
| dest         | string        | The path of the destination file  | IN   | yes      |
| transferInfo | FileTransfer  | A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it) | OUT  | yes      |
| options      | CpFileOptions | the copy options  | IN   | 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |

| Name      | Description                                 |
|-----------|---|
| VISHNU_OK | The service has been performed succesfully. |

**Signature**

int **vishnu::copyAsyncFile**(const string& sessionKey, const string& src, const string& dest, FileTransfer& transferInfo, const CpFileOptions& options);

**5.1.13 Function FMS::moveFile****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type          | Description  | Mode | Required |
|------------|---------------|--|------|----------|
| sessionKey | string        | The session key  | IN   | yes      |
| src        | string        | The source file to move following the pattern [host:]file path | IN   | yes      |
| dest       | string        | The path of the destination file                               | IN   | yes      |
| options    | CpFileOptions | The move command options                                       | IN   | yes      |

**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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

**Signature**

int **vishnu::moveFile**(const string& sessionKey, const string& src, const string& dest, const CpFileOptions& options);

**5.1.14 Function FMS::moveAsyncFile****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter    | Type          | Description   | Mode | Required |
|--------------|---------------|---|------|----------|
| sessionKey   | string        | The session key   | IN   | yes      |
| src          | string        | The source file to move following the pattern [host:]file path  | IN   | yes      |
| dest         | string        | The path of the destination file  | IN   | yes      |
| transferInfo | FileTransfer  | A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it) | OUT  | yes      |
| options      | CpFileOptions | The transfer 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

### Signature

```
int vishnu::moveAsyncFile(const string& sessionKey, const string& src, const string& dest, FileTransfer& transferInfo, const CpFileOptions& options = CpFileOptions());
```

## 5.1.15 Function FMS::stopFileTransfer

### Access

This function can be used by any VISHNU user

### Parameters

| Parameter  | Type                | Description                            | Mode | Required |
|------------|---------------------|--|------|----------|
| sessionKey | string              | The session key                        | IN   | yes      |
| options    | StopTransferOptions | The stop file transfer command options | IN   | no       |

### Description

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

### Return Value

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

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |

### Signature

```
int vishnu::stopFileTransfer(const string& sessionKey, const StopTransferOptions& options = StopTransferOptions());
```

### 5.1.16 Function FMS::listFileTransfers

#### Access

This function can be used by any VISHNU user

#### Parameters

| Parameter        | Type                     | Description            | Mode | Required |
|------------------|--------------------------|------------------------|------|----------|
| sessionKey       | string                   | The session key        | IN   | yes      |
| fileTransferList | <b>FileTransferList</b>  | The file transfer list | OUT  | yes      |
| options          | <b>LsTransferOptions</b> | The filter options     | IN   | no       |

#### Description

The listFileTransfers() function displays the history of all file transfers submitted by User.

#### Return Value

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

| Name                          | Description  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id ins unknown                                    |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed succesfully.                    |



**Signature**

```
int vishnu::listFileTransfers(const string& sessionKey, FileTransferList& fileTransferList, const LsTransferOptions& options = LsTransferOptions());
```

**5.1.17 Function FMS::getFileInfo****Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter  | Type     | Description  | Mode | Required |
|------------|----------|--|------|----------|
| sessionKey | string   | The session key                                    | IN   | yes      |
| path       | string   | The file whose inode information will be displayed | IN   | yes      |
| filesinfo  | FileStat | The inode information                              | OUT  | yes      |

**Description**

The getFileInfo() 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  |
|-------------------------------|--|
| ERRCODE_CONFIGNOTFOUND        | Undefined configuration parameter                              |
| ERRCODE_DB_ERROR              | Vishnu not available (Database error)                          |
| ERRCODE_DIET                  | Vishnu not available (Service bus failure)                     |
| ERRCODE_INVALID_PARAM         | an option or a parameter provided is invalid for this service. |
| ERRCODE_INVALID_PATH          | The path provided is invalid.                                  |
| ERRCODE_NO_ADMIN              | The user is not an administrator                               |
| ERRCODE_RUNTIME_ERROR         | Runtime error  |
| ERRCODE_SESSIONKEY_EXPIRED    | The session key has expired. The session is closed.            |
| ERRCODE_SESSIONKEY_NOT_FOUND  | The session key is unrecognized                                |
| ERRCODE_UNDEFINED             | Internal Error: Undefined exception                            |
| ERRCODE_UNKNOWN_FILE_TRANSFER | The transfer id is unknown                                     |
| ERRCODE_UNKNOWN_LOCAL_ACCOUNT | The local account is unknown                                   |
| ERRCODE_UNKNOWN_MACHINE       | The machine id is unknown                                      |
| VISHNU_OK                     | The service has been performed successfully.                   |

**Signature**

```
int vishnu::getFileInfo(const string& sessionKey, const string& path, FileStat& filesinfo);
```

**5.2 Data types definitions****Class FMS::CpFileOptions Content**

| Name        | Type            | Description   |
|-------------|-----------------|---|
| isRecursive | boolean         | It specifies when the copy is recursive (case of directory) or not. |
| trCommand   | TransferCommand | the command to use to perform file transfer.                        |

**Class FMS::FileStat Content**



| Name  | Type     | Description  |
|-------|----------|--|
| path  | string   | The path of the file                                 |
| owner | string   | The name of the owner of the file                    |
| group | string   | The group name of the owner                          |
| perms | int      | The protection of the file                           |
| uid   | long     | The user identifier of the owner                     |
| gid   | long     | The group identifier of the owner                    |
| size  | long     | The size of the file in bytes                        |
| atime | long     | The time of last access                              |
| mtime | long     | The time of last modification                        |
| ctime | long     | The time of the last change of the inode information |
| type  | FileType | The file type  |

**Class FMS::FileTransfer Content**

| Name                 | Type            | Description  |
|----------------------|-----------------|--|
| transferId           | string          | The file transfer identifier                           |
| status               | Status          | The file transfer status                               |
| userId               | string          | The user identifier                                    |
| clientMachineId      | string          | The client machine identifier                          |
| sourceMachineId      | string          | The source machine identifier                          |
| destinationMachineId | string          | The destination machine identifier                     |
| sourceFilePath       | string          | The source file path                                   |
| destinationFilePath  | string          | The destination file path                              |
| size                 | long            | The size of the file                                   |
| start_time           | long            | The start time of the file transfer                    |
| trCommand            | TransferCommand | The command used for the file transfer (scp or rsync)  |
| errorMsg             | string          | The eventual error message if the file transfer failed |

**Class FMS::FileTransferList Content**

| Name          | Type             | Description             |
|---------------|------------------|-------------------------|
| fileTransfers | List of FileType | list of file transfers. |

**Class FMS::HeadOfFileOptions Content**

| Name  | Type | Description               |
|-------|------|---------------------------|
| nline | int  | the number of line to get |

**Class FMS::LsDirOptions Content**

| Name       | Type    | Description  |
|------------|---------|--|
| longFormat | boolean | It specifies the long display format (all available file informations) |
| allFiles   | boolean | Allows to display all files including hidden files                     |

**Class FMS::LsTransferOptions Content**

| Name          | Type   | Description  |
|---------------|--------|--|
| transferId    | string | a given transfer id  |
| fromMachineId | string | the machine that is the source of the file transfer                  |
| userId        | string | allows the admin to list file transfers initiated by a specific user |
| status        | Status | the file transfer status   |

**Class FMS::RmFileOptions Content**

| Name        | Type    | Description   |
|-------------|---------|---|
| isRecursive | boolean | It specifies when the remove command is recursive (case of directory) or not. |

**Class FMS::StopTransferOptions Content**

| Name          | Type   | Description  |
|---------------|--------|--|
| transferId    | string | a given transfer id  |
| fromMachineId | string | the machine that is the source or destination of the file transfer |
| userId        | string | allows an admin to stop file transfers of a specific user          |

**Class FMS::StringList Content**

| Name    | Type           | Description                  |
|---------|----------------|------------------------------|
| strings | List of string | the dynamic list of strings. |

**Enumeration FMS::FileType Type**

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

**Enumeration FMS::Status Type**

| Name       | Value |
|------------|-------|
| INPROGRESS | 0     |
| COMPLETED  | 1     |
| CANCELLED  | 2     |
| FAILED     | 3     |
| UNDEFINED  | 4     |

**Enumeration FMS::TransferCommand Type**

| Name      | Value |
|-----------|-------|
| SCP       | 0     |
| RSYNC     | 1     |
| UNDEFINED | 2     |