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

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>D1.1c - VISHNU A.P.I. specifications | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé | February 10, 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, UMS::AddMachineOptions and UMS::UpdateUserOptions classes and UMS::SessionStateType.<br>Added function UMS::reconnect, UMS::StatusType.<br>Modified UMS::restoreConfiguration and UMS::Configuration class | SysFera |

# Contents

# 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 or 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 |
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU | 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 |
| NOT_AUTHENTICATED | Unknown user |
| UNKNOWN_CLOSURE_MODE | The name of the closure mode is unknown |
| INCORRECT_TIMEOUT | The value of the timeout is incorrect (negative or higher than the TIMEOUT treshold) |
| UNKNOWN_USERID | The userId is unknown |
| NO_ADMIN | The user is not an administrator |
| INCORRECT_PASSWORD_SIZE | The size of the password is incorrect |

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

**Signature**

int **connect**(const string& userId, const string& password, string& sessionKey, 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 |
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU | OUT | yes |

**Description**

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

**Return Value**

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

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully |
| NOT_AUTHENTICATED | Unknown user |
| INCORRECT_USERID_SIZE | The size of the userId is incorrect |
| INCORRECT_PASSWORD_SIZE | The size of the password is incorrect |
| SESSION_INCOMPATIBILITY | This session identifier is incompatible with the authenticated user |
| UNKNOWN_SESSION_ID | The session Id is unknown |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| UNKNOWN_MACHINE_FOR_SESSION | This session has been opened with another machine |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.3 Function UMS::addVishnuUser

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

The addVishnuUser() function adds a new VISHNU user

**Return Value**

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

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

**Signature**

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

### 2.1.4 Function UMS::updateUser

**Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| sessionKey | string | The sessionKey is the 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 |
| NO_ADMIN | The user is not an administrator |
| UNKNOWN_USERID | The userId is unknown |
| INVALID_MAIL_ADRESS | The mail adress is invalid |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |

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

**Signature**

int **updateUser**(const string& sessionKey, const 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 |
| UNKNOWN_USERID | The userId is unknown |
| NO_ADMIN | The user is not an administrator |
| INCORRECT_USERID_SIZE | The size of the userId is incorrect |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **deleteUser**(const string& 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 identfied by the session key

**Return Value**

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

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully |
| COMMAND_RUNNING | Command(s) is/are running |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.7 Function UMS::changePassword

**Access**

This function can be used by any VISHNU user

**Parameters**

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

**Description**

The changePassword() function changes the password

**Return Value**

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

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

**Signature**

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

### 2.1.8 Function UMS::resetPassword

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

The resetPassword() function resets the password of a user

**Return Value**

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

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

**Signature**

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

### 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 |

**Description**

The addLocalAccount() function adds a new local user configuration

**Return Value**

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

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully |
| LOCAL_ACCOUNT_EXIST | The local account already exists for the given user on the given machine |
| USERID_REQUIRED | The userId must to be defined |
| MACHINEID_REQUIRED | The machineId must to be defined |
| UNKNOWN_USERID | The userId is unknown |

| Name | Description |
|---|---|
| UNKNOWN_MACHINE | The machineId is unknown |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.10 Function UMS::updateLocalAccount

**Access**

This function can be used by any VISHNU user

**Parameters**

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

**Signature**

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

### 2.1.11 Function UMS::deleteLocalAccount

**Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| 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 |
| UNKNOWN_LOCAL_ACCOUNT | The local configuration for the given user on the given machine is unknown |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **deleteLocalAccount**(const string& 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 |
| filePath | string | The filePath is the path of the file in which the VISHNU configuration will be saved | IN | yes |
| configuration | Configuration | The configuration is an object which encapsulates the configuration description | OUT | yes |

**Description**

The saveConfiguration() function saves the configuration of VISHNU

**Return Value**

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

| Name | Description |
|---|---|
| VISHNU_OK | The service was performed successfully |
| NO_ADMIN | The user is not an administrator |
| SAVE_CONFIG_ERROR | A problem occurs during the configuration saving |

| Name | Description |
|---|---|
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **saveConfiguration**(const string& sessionKey, const string& filePath, 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 |
| NO_ADMIN | The user is not an administrator |
| RESTORE_CONFIG_ERROR | A problem occurs during the configuration restoring |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **restoreConfiguration**(const string& sessionKey, const 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 | IN | yes |

**Description**

The addMachine() function adds a new machine in VISHNU

**Return Value**

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

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

**Signature**

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

### 2.1.15 Function UMS::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 |
| MACHINEID_REQUIRED | The machineId must to be defined |
| UNKNOWN_MACHINE | The machineId is unknown |
| NO_ADMIN | The user is not an administrator |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **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 |
| NO_ADMIN | The user is not an administrator |
| UNKNOWN_MACHINE | The machineId is unknown |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.17 Function UMS::listLocalAccount

**Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| sessionKey | string | The sessionKey is the encrypted identifier of the session generated by VISHNU | IN | yes |
| listLocalAcct | ListLocalAccounts | listLocalAccount is the list of the local user configuations | 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 listLocalAccount() function lists the local user configurations

**Return Value**

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

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

**Signature**

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

### 2.1.18 Function UMS::listMachine

**Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| sessionKey | string | The sessionKey is the 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 listMachine() function lists the machines in which the local user configurations are defined for the given user

**Return Value**

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

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

**Signature**

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

**Signature**

int **listHistoryCmd**(const string& sessionKey, 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 |
| NO_ADMIN | The user is not an administrator |

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

**Signature**

int **listOptions**(const string& sessionKey, 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 |
|---|---|
| NO_ADMIN | The user is not an administrator |
| UNKNOWN_USERID | The userId is unknown |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| INCORRECT_DATE_OPTION | The date option is incorrect |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **listUsers**(const string& sessionKey, 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 |
| UNKNOWN_USERID | The userId is unknown |
| UNKNOWN_SESSION_OPTION | The name of the session option is unknown |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| INCORRECT_DATE_OPTION | The date option is incorrect |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

int **listSessions**(const string& sessionKey, 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 |
| NO_ADMIN | The user is not an administrator |
| UNKNOWN_OPTION | the name of the option is unknown |

| Name | Description |
|---|---|
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.24 Function UMS::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 |
| UNKNOWN_OPTION | the name of the option is unknown |
| SESSIONKEY_EXPIRED | The sessionKey is expired. The session is closed. |
| SESSIONKEY_NOT_FOUND | The sessionKey is unrecognized |
| DB_ERROR | A problem occurs with the database |
| UMS_SERVER_NOT_AVAILABLE | The server UMS is not available |

**Signature**

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

### 2.1.25 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 |
| INCORRECT_CFG_PATH | The configuration file path is incorrect |

**Signature**

int **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 |
| VISHNU_FINALIZE_ERROR | An error occurs during vishnuFinalize |

**Signature**

int **vishnuFinalize**();

### 2.1.27 Function UMS::restore

**Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| file | string | file containing a sql script to restore the database | IN | yes |

**Description**

The restore() function initializes VISHNU environment for tests

**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 |

**Signature**

int **restore**(const string& file);

## 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) |

**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 |

**Class UMS::ConnectOptions Content**

| Name | Type | Description |
|---|---|---|
| closePolicy | SessionCloseType | is an option for closing session automatically |
| sessionInactivityDelay | int | The sessionInactivityDelay is the maximum delay in seconds between two API commands when the CLOSE_ON_TIMEOUT option is set |
| substituteUserId | string | is an option which allows an admin to open a session as if he/she was a specific user identified by his/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 Command | 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 LocalAccount | 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 Machine | 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 OptionValue | 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 |
| language | string | represents the language in which the description of the machine has been done |
| status | StatusType | represents the status of the machine |

**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::PrivilegeType Type**

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

| Name | Value |
|---|---|
| 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 sessionKey is the identifier of the session generated by VISHNU | IN | yes |
| machineId | string | Is the id of the machine where the job must be submitted | IN | yes |
| scriptFilePath | string | The path to the file containing the characteristics (job command, and batch scheduler directive required or optional) of the job to submit. | IN | yes |
| jobId | string | Is the returned id of the submitted job | OUT | yes |
| jobPath | string | Is the path to the file containing job characteristics | OUT | yes |
| options | SubmitOptions | Is an instance of the class SubmitOptions. Each optionnal value is associated to a set operation (e.g: setNbCpu(...)) in the class SubmitOptions. If no set operation is not called on the instance object options, the job is submitted with the options defined in the scriptFilePath. Otherewise 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 occurance of an optionnal value takes precedence over earlier occurance. | IN | no |

**Description**

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

**Return Value**

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

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

**Signature**

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

### 3.1.2 Function TMS::getJobInfo

**Access**

This function can be used by any VISHNU user

**Parameters**

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

**Description**

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

**Return Value**

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

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

| Name | Description |
|---|---|
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Signature**

int **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 sessionKey is the identifier of the session generated by VISHNU | IN | yes |
| machineId | string | Is the id of the machine to get the jobs progression. | IN | yes |
| progress | Progression | Is the object containing jobs progression information | OUT | yes |
| options | ProgressOptions | Is an object containing the available options jobs for progression . | IN | no |

**Description**

The getJobProgress() function 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. |
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_SUBMIT_SERVICE_NOT_AVAILABLE | Indicates that the service to perform the submit operation is not found. |
| DB_ERROR | A problem occurs with the database |

**Signature**

int **getJobProgress**(const string& sessionKey, const string& machineId, Progression& progress, 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 sessionKey is the identifier of the session generated by VISHNU | IN | yes |
| machineId | string | Machine hash key | IN | yes |
| listofQueues | ListQueues | The list of queues | OUT | yes |

**Description**

The listQueues() function gets queues information

**Return Value**

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

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

**Signature**

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

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

**Description**

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

**Return Value**

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

| Name | Description |
|---|---|
| TMS_BATCH_SCHEDULER_ERROR | Indicates an error caused by the underlying batch scheduler |
| TMS_INVALID_RESPONSE | Indicates that the implementation produced a response that does not match the criteria defined by the specification. |

| Name | Description |
|---|---|
| TMS_INVALID_REQUEST | Indicates that the request is not valid. |
| TMS_INVALID_SESSION_KEY | The session key is not valid to perform the service. |
| TMS_PERMISSION_DENIED | Indicates the requested operation is not allowed for provided user. |
| VISHNU_OK | The service was performed successfully. |
| TMS_SERVER_NOT_AVAILABLE | Indicates that the task management server is not available. |
| TMS_UNKNOWN_BATCH_SCHEDULER_TYPE | Indicates that the batch scheduler type is not known. |
| TMS_UNKNOWN_MACHINE | The machine is not known. |
| DB_ERROR | A problem occurs with the database |

**Signature**

int **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 sessionKey is the identifier of the session generated by VISHNU | IN | yes |
| machineId | string | Machine hash key | IN | yes |
| jobId | string | The Id of the job | IN | yes |
| outputPath | string | The path of the file containinig the output result of the job | OUT | yes |
| errorPath | string | The path of the file containinig the errors that has been occurred during the execution of the job | OUT | yes |

**Description**

The getJobOutPut() function gets outputPath and errorPath of a job from its id

**Return Value**

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

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

**Signature**

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

### 3.1.7 Function TMS::getAllJobsOutPut

**Access**

This function can be used by any VISHNU user

**Parameters**

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

**Description**

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

**Return Value**

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

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

**Signature**

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

### 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 sessionKey is the identifier of the session generated by VISHNU | IN | yes |
| machineId | string | Machine hash key | IN | yes |
| jobId | string | The Id of the job | IN | yes |
| infoMsg | string | The information message | OUT | yes |

**Description**

The cancelJob() function cancels a job from its id

**Return Value**

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

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

**Signature**

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

### 3.1.9 Function TMS::setMachineEnv

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

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

**Return Value**

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

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

**Signature**

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

### 3.1.10 Function TMS::setMachineRefreshPeriod

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

The setMachineRefreshPeriod() function sets the refresh period of output and error files contents

**Return Value**

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

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

**Signature**

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

## 3.2 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 occured 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. |

| Name | Type | Description |
|---|---|---|
| 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. |
| outputPath | string | Is the path to the job output results. |
| errorPath | string | Is the path to the file containing errors occured during job's execution. |

**Class TMS::ListJobResults Content**

| Name | Type | Description |
|---|---|---|
| nbJobs | string | Is the number of jobs. |
| Results | List of JobResult | 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 Job | Is a list of job information (jobId, jobName, ...). |

**Class TMS::ListJobsOptions Content**

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

**Class TMS::ListQueues Content**

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

**Class TMS::ProgressOptions Content**

| Name | Type | Description |
|---|---|---|
| jobId | string | Represents the id of the job that the user wants to see the progression of. |

| Name | Type | Description |
|---|---|---|
| jobOwner | string | Represents 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 | double | 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 Cups 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. |
| priority | JobPriority | Assigns priority of the job. |
| queue | string | Assigns the queue or class of the job. |
| wallTime | int | The maximum wall-clock time during which the job can run. |
| memory | int | 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 |
|---|---|
| VERY_LOW | 100 |
| LOW | 200 |
| NORMAL | 300 |
| HIGH | 400 |
| VERY_HIGH | 500 |

**Enumeration TMS::JobStatus Type**

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

**Enumeration TMS::QueuePriority Type**

| Name | Value |
|---|---|
| VERY_LOW | 0 |
| LOW | 1 |
| NORMAL | 2 |
| HIGH | 3 |
| VERY_HIGH | 4 |

**Enumeration TMS::QueueStatus Type**

| Name | Value |
|---|---|
| STARTED | 0 |
| RUNNING | 1 |
| NOT_STARTED | 2 |
| NOT_AVAILABLE | 3 |

# 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 key of the session of the user that submits the command (current session key) | IN | yes |
| oldSessionKey | string | The key of the session to export (session has ended) | IN | yes |
| filename | string | The path of the output file containing the Vishnu shell commands | INOUT | yes |

**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 |
| DB_ERROR | The database generated an error |
| INVALID_PARAMETER | If a parameter is invalid |

**Signature**

int **exportCommands**(const string& sessionKey, const string& oldSessionKey, string& filename);

### 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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine | IN | yes |
| metricType | MetricType | Type of metric | IN | yes |
| metricValue | Metric | Value of the metric | OUT | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **getMetricCurrentValue**(const string& sessionKey, const string& machineId, const MetricType& metricType, Metric& metricValue);

### 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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine | IN | yes |
| startTime | long | Start time of metric history | IN | yes |
| endTime | long | End time of metric history | IN | yes |
| metricType | MetricType | Type of metric | IN | yes |
| metricValues | ListMetric | List of metric values | OUT | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **getMetricHistory**(const string& sessionKey, const string& machineId, const long& startTime, const long& endTime, const MetricType& metricType, ListMetric& metricValues);

### 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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine the user wants the running processes | IN | yes |
| process | ListProcesses | The list of the Vishnu processes on the machine | OUT | yes |

**Description**

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

**Return Value**

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

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

**Signature**

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

### 4.1.5 Function IMS::setSystemInfo

**Access**

This function can be used by ADMIN users only

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| sessionKey | string | The key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine | 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 |

| Name | Description |
|---|---|
| DB_ERROR | The database generated an error |
| INVALID_PARAMETER | If a parameter is invalid |

**Signature**

int **setSystemInfo**(const string& sessionKey, const string& machineId, 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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine | IN | yes |
| thresholdType | MetricType | The type of the metric to set | IN | yes |
| value | double | The threshold value | IN | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **setSystemThreshold**(const string& sessionKey, const string& machineId, const MetricType& thresholdType, const double& value);

### 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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine | IN | yes |
| type | MetricType | The threshold type desired | IN | yes |
| value | double | The threshold value | OUT | yes |

**Description**

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

**Return Value**

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

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

**Signature**

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

### 4.1.8 Function IMS::defineUserIdentifier

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

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

**Return Value**

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

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

**Signature**

int **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 key of the session of the user that submits the command | IN | yes |
| format | string | The new format to use | IN | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **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 key of the session of the user that submits the command | IN | yes |
| format | string | The new format to use | IN | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **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 key of the session of the user that submits the command | IN | yes |
| format | string | The new format to use | IN | yes |

**Description**

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

**Return Value**

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

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

**Signature**

int **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 key of the session of the user that submits the command | IN | yes |
| machineId | string | The id of the machine to stop | IN | yes |
| loadShedType | LoadShedType | 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 load sheds 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 |
| COMPONENT_ERROR | If a component is unavailable |
| DB_ERROR | The database generated an error |
| INVALID_PARAMETER | If a parameter is invalid |

**Signature**

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

### 4.1.13 Function IMS::setUpdateFrequency

**Access**

This function can be used by ADMIN users only

**Parameters**

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

**Description**

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

**Return Value**

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

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

**Signature**

int **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 key of the session of the user that submits the command | IN | yes |
| freq | int | Frequency the data are updated, in second | OUT | yes |

**Description**

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

**Return Value**

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

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

**Signature**

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

## 4.2 Data types definitions

**Class IMS::ListMetric Content**

| Name | Type | Description |
|---|---|---|
| metric | List of Metric | The metrics of the list |

**Class IMS::ListProcesses Content**

| Name | Type | Description |
|---|---|---|
| processName | List of string | The Vishnu processes of the list |

**Class IMS::Metric Content**

| Name | Type | Description |
|---|---|---|
| type | MetricType | The type of the metric |
| value | double | The value of the metric |
| time | int | The timestamp the metric had the value |

**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) |

**Enumeration IMS::LoadShedType Type**

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

**Enumeration IMS::MetricType Type**

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

# 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 |
| mode | int | The file access permissions in octal numeral system | IN | no |

**Description**

The createFile() function creates files on remote machines.

**Return Value**

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

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

**Signature**

int **createFile**(const string& sessionKey, const string& path, const int& mode = 644);

### 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 |
| mode | int | the new directories permission access in octal numeral system | IN | no |

**Description**

The createDir() function creates directories on remote machines.

**Return Value**

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

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

**Signature**

int **createDir**(const string& sessionKey, const string& path, const int& mode = 755);

### 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 |

**Description**

The removeFile() function removes files from remote hosts.

**Return Value**

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

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

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

**Signature**

int **removeFile**(const string& sessionKey, const string& path);

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

**Signature**

int **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 |
| path | string | The file/directory following the pattern [host:]file path | IN | yes |
| group | string | the new group owner of file/directory | 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 |
|---|---|
| VISHNU_OK | The service has been performed succesfully. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| DB_ERROR | A problem occurs with the database. |

**Signature**

int **chGrp**(const string& sessionKey, const string& path, const string& group);

### 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 |
| path | string | The file/directory following the pattern [host:]file path | IN | yes |
| mode | int | the access rigths of file/directory in octal system. | 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 |
|---|---|
| VISHNU_OK | The service has been performed succesfully. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| DB_ERROR | A problem occurs with the database. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |

**Signature**

int **chMod**(const string& sessionKey, const string& path, const int& mode);

### 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 |
| nLine | int | Number of lines to display | 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 |
|---|---|
| VISHNU_OK | The service has been performed succesfully. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| DB_ERROR | A problem occurs with the database. |

**Signature**

int **headOfFile**(const string& sessionKey, const string& path, string& fileContent, const int& nLine = 10);

### 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 |
| nLine | int | number of lines to display | 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 |
|---|---|
| VISHNU_OK | The service has been performed succesfully. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| DB_ERROR | A problem occurs with the database. |

**Signature**

int **tailOfFile**(const string& sessionKey, const string& path, string& fileContent, const int& nLine = 10);

### 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 |
|---|---|
| VISHNU_OK | The service has been performed succesfully. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| DB_ERROR | A problem occurs with the database. |

**Signature**

int **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 |
| 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 | no |

**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 |
|---|---|
| DB_ERROR | A problem occurs with the database. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| VISHNU_OK | The service has been performed succesfully. |

**Signature**

int **listDir**(const string& sessionKey, const string& path, StringList& dirContent, const LsDirOptions& options = LsDirOptions());

### 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 | CopyFileOptions | the copy options | IN | no |

**Description**

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

**Return Value**

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

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

**Signature**

int **copyFile**(const string& sessionKey, const string& src, const string& dest, const CopyFileOptions& options = CopyFileOptions());

### 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 |
| thrId | long | A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it) | OUT | yes |
| options | CopyFileOptions | the copy options | IN | no |

**Description**

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

**Return Value**

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

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

**Signature**

int **copyAsyncFile**(const string& sessionKey, const string& src, const string& dest, long& thrId, const CopyFileOptions& options = CopyFileOptions());

### 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 | MvFileOptions | The move command options | IN | no |

**Description**

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

**Return Value**

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

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

**Signature**

int **moveFile**(const string& sessionKey, const string& src, const string& dest, const MvFileOptions& options = MvFileOptions());

### 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 |
| thrId | long | The file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it) | OUT | yes |
| options | MvFileOptions | The mv command options | IN | no |

**Description**

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

**Return Value**

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

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

**Signature**

int **moveAsyncFile**(const string& sessionKey, const string& src, const string& dest, long& thrId, const MvFileOptions& options = MvFileOptions());

### 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 |
|---|---|
| DB_ERROR | A problem occurs with the database. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| VISHNU_OK | The service has been performed succesfully. |

**Signature**

int **stopFileTransfer**(const string& sessionKey, const StopTransferOptions& options = StopTransferOptions());

### 5.1.16 Function FMS::listFileTransferStatus

**Access**

This function can be used by any VISHNU user

**Parameters**

| Parameter | Type | Description | Mode | Required |
|---|---|---|---|---|
| sessionKey | string | The session key | IN | yes |
| options | LsTrStatusOptions | The filter options | IN | no |

**Description**

The listFileTransferStatus() function displays the status 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 |
|---|---|
| DB_ERROR | A problem occurs with the database. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |
| INVALID_SESSION_KEY | Invalid provided session key. |
| VISHNU_OK | The service has been performed succesfully. |

**Signature**

int **listFileTransferStatus**(const string& sessionKey, const LsTrStatusOptions& options = LsTrStatusOptions());

### 5.1.17 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 |
| options | LsTransferOptions | 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 |
|---|---|
| DB_ERROR | A problem occurs with the database. |
| FMS_INVALID_CFG_FILE | Option unknown in the config file. |
| FMS_INVALID_PARAMETER | an option or a parameter provided is invalid for this service. |
| FMS_INVALID_PATH | The path provided is invalid. |
| FMS_PERMISSION_DENIED | User does not have permission access. |
| FMS_SERVER_UNAVAILABLE | The FMS server is unavailable. |
| INVALID_MACHINE_ID | Invalid provided machine identifier. |

| Name | Description |
|---|---|
| INVALID_SESSION_KEY | Invalid provided session key. |
| VISHNU_OK | The service has been performed succesfully. |

**Signature**

int **listFileTransfers**(const string& sessionKey, const LsTransferOptions& options = LsTransferOptions());

### 5.1.18 Function FMS::getFilesInfo

**Access**

This function can be used by any VISHNU user

**Parameters**

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

**Description**

The getFilesInfo() function displays the information of files.

**Return Value**

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

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

**Signature**

int **getFilesInfo**(const string& sessionKey, const string& pathList, FileStat& filesinfo);

## 5.2 Data types definitions

**Class FMS::CopyFileOptions 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 |
| 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::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::LsTrStatusOptions 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 |

**Class FMS::LsTransferOptions Content**

| Name | Type | Description |
|---|---|---|
| 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::MvFileOptions Content**

| Name | Type | Description |
|---|---|---|
| trCommand | TransferCommand | the command to use to perform file transfer. |

**Class FMS::StopTransferOptions 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 an admin to stop file transfers of a specific user |

**Class FMS::StringList Content**

| Name | Type | Description |
|---|---|---|
| listOfStrings | 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 |

**Enumeration FMS::TransferCommand Type**

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