VISHNU User Manual

COLLABORATORS			
	TITLE : VISHNU User Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, Ibrahima Cissé, and Rodrigue Chakode	June 21, 2012	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1	08/03/2011	First version of the VISHNU user manual which concerns only the UMS package.	SysFera
2	03/05/2011	Add of details concerning the TMS package.	SysFera
3	15/06/2011	Add of details concerning the IMS and FMS package.	SysFera
4	28/06/2011	Add of TMS generic script example.	SysFera
5	18/07/2011	Add of the CLI examples and fix some mistakes..	SysFera
6	11/08/2011	Add of SLURM batch scheduler	SysFera
7	23/08/2011	Add the only one local account per machine warning. Add a reference to VISHNU_API	SysFera

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
8	14/12/2011	Add a section dedicated of the VISHNU Job Output Environment Variables. Add of other syntaxes for VISHNU generic script. Update of vishnu submit job reference, vishnu create dir reference. Add of vishnu_current_session_id a new command line reference. Add the use of the .netrc file of the connection and the reconnection.	SysFera
9	16/12/2011	Add a section dedicated of the Configuration of ssh keys required for TMS.	SysFera
10	10/01/2012	Updates of TMS: Modified decription of vishnu_submit_job command (to take into account automatic submission). Extended vishnu_submit_job options (added load criterion option for automatic submission, added option to select a queue automatically). Modified decription of vishnu_list_jobs command (to take into account listing of jobs on all machines). Extended vishnu_list_jobs options (added multipleStatus option for combination of several job states, added option to list all jobs submitted by the underlying batch scheduler (VISHNU jobs and jobs submitted out of VISHNU). Modified decription of submitJob C++ and Python API function (to take into account automatic submission). Modified decription of listJobs C++ and Python API function (to take into account listing of jobs on all machines).	SysFera
11	06/03/2012	Add of LSF batch scheduler.	SysFera
12	08/03/2012	Add of VISHNU commands for LDAP support and for connection and reconnection using multiples VISHNU accounts.	SysFera
13	11/04/2012	Add of Grid Engine batch scheduler.	SysFera
14	22/05/2012	Add FAQ section.	SysFera

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
15	21/06/2012	Add a section for the service vishnu_set_ssh_key in UMS that allows to configure ssh keys for remote accounts.	SysFera
16	28/06/2012	Add a section for the copy/move commands in FMS with the client out of the DNS.	SysFera

Contents

Chapter 1

Document presentation

1.1 Document objectives

This documents is a quick start guide of VISHNU software for users. The main objective of this document is to describe the VISHNU installation procedure and the way to use it.

1.2 Document structure

- Chapter 1 presents the document structure.
- Chapter 2 describes the VISHNU software (installation procedure, usage description and troubleshooting).
- Chapter 3, Chapter 4, Chapter 5 and Chapter 6 contain the VISHNU commands reference respectively for UMS, TMS, FMS and IMS package.
- Chapter 7, Chapter 8, Chapter 9 and Chapter 10 contain the C++ API reference respectively for UMS, TMS, FMS and IMS package.
- Chapter 11, Chapter 12, Chapter 13 and Chapter 14 contain the Python API reference respectively for UMS, TMS, FMS and IMS package.

1.3 References

- [D1.1b]: VISHNU "Spécifications techniques des besoins"
- [DIETMAN]: DIET User's Manual v2.8 (available with the DIET distribution at http://graal.ens-lyon.fr/~diet)
- [VISHNU_API] VISHNU API : Document that contain the VISHNU API description and all the datatypes used..

Chapter 2

Installation and usage

The VISHNU software is based on SysFera-DS which is an open-source middleware developped by SysFera. VISHNU is primarly designed to facilitate the access to high-performance computing resources by providing the following services:

- User management services (UMS): authentication and session management.
- Information management services (IMS): monitoring and control services.
- Tasks management services (TMS): submission of tasks (jobs) on computing resources.
- File management services (FMS): display and transfer of files between storage resources.

2.1 Installation procedure of the clients

This section details the main steps of the installation process for the clients, including the installation requirements [D1.1b]. VISHNU is based on SysFera-DS software which must be installed before.

Installation requirements:

- GCC V4.4.3
- CMAKE V2.6
- OMNIORB 4.1.4
- SYSFERA-DS V2.8 (available at: http://graal.ens-lyon.fr/DIET/)
- BOOST V1.46.1
- PYTHON V2.5
- JAVA V1.6
- SWIG V1.3
- LIBCRYPT

Installation procedure:

2.1.1 From sources

- Download the VISHNU install sources
- Decompress it and go to the vishnu directory
- Create a build directory and run CMake as follows:

  > mkdir build

  > cd build

  If your install directory is for example: /opt/vishnu

  > cmake -DCLIENT_ONLY=ON -DCMAKE_INSTALL_PREFIX=/opt/vishnu ..

  > make && make install

- The module for each client can be built using the '-DCOMPILE_*MS=ON' flag. These are some commande line examples on how to build the clients
  - **To compile the UMS package :**
    > cmake -DCOMPILE_UMS=ON -DCMAKE_INSTALL_PREFIX=/opt/vishnu ..
  - **To compile the TMS package :**
    > cmake -DCOMPILE_UMS=ON -DCOMPILE_TMS=ON -DCMAKE_INSTALL_PREFIX=/opt/vishnu ..
  - **To compile the FMS package :**
    > cmake -DCOMPILE_UMS=ON -DCOMPILE_FMS=ON -DCMAKE_INSTALL_PREFIX=/opt/vishnu ..
  - **To compile the IMS package :**
    > cmake -DCOMPILE_UMS=ON -DCOMPILE_TMS=ON -DCOMPILE_FMS=ON -DCOMPILE_IMS=ON -DCMAKE_INSTALL_PREFIX=/opt/vishnu ..

2.1.2 From binaries package

Assuming the dependencies are installed (omniorb, diet, boost). Download the vishnu-client_*_i386.deb, where * stands for the VISHNU version and install it (dpkg -i vishnu_client_*_i386.deb). Then, if the DIET hierarchy is already running and accessible from the client, one can uses VISHNU. All the services are available in the debian packages.

2.2 Software usage description

VISHNU is composed of 4 main packages, one that deals with the users and the machine (UMS), one that deals with the batch schedulers (TMS), one that deals with the files management (FMS) and one that handle the information of the system (IMS). These clients can be installed alltogether or just one or two. Please contact your VISHNU admin to know the corresponding servers you can have before installing the client. It is important to note that the client can be installed without the server being on the platform, the call will end up with an exception and the message "Vishnu not available". Below each package will be describe to show the services it offers to the user.

More information about the datatypes can be found in the [VISHNU_API] document.

WARNING: The lists are INOUT parameters, the results are appended, they do not overwrite the existing list. Moreover, the get function on the lists do not check the bounds, it is like using the [] operator.

2.2.1 UMS package

2.2.1.1 User account creation

The first step to access VISHNU is to request a new account to a VISHNU administrator. The only information required to create a new account is your full name and email address. You will automatically receive an email containing your userId and password.

2.2.1.2 Connection to VISHNU

To connect, use the **vishnu_connect** command in the shell terminal (all bourne shell are supported). The password received by email is temporary and must be changed at the first connection by using the **vishnu_change_password** command. It is also possible to make connection by using the .netrc file. Indeed when the userId and the password are empty, with **vishnu_connect** without parameters, the system automatically gets the login and the password from the .netrc file. A vishnu account on the .netrc file can be defined as follows:

machine vishnu

login toto

password pwd

The values toto and pwd are respectively the userId and the password of a user registered in vishnu and the machine must be named **vishnu**.

It is also possible to define a serie of VISHNU accounts on the .netrc file. The System will try them each in turn until one which will allow to make a connection. As the .netrc file, the **vishnu_connect_m** command allows to give successively several couples login/password.

2.2.1.3 Reconnection to VISHNU

Reconnection is done using the **vishnu_reconnect** command. This command allows using an existing session that was previously opened but not closed. It makes it possible to simultaneously use the same session in different shell terminals. A session is what authentifies a user once he has connected. The user does not need password or username when authenticated to use vishnu. Moreover, the session contains the commands made by a user, so he can retrieve the sequence of commands made in a previous work session. As previously defined for the connection, the .netrc file can be used for reconnection. The **vishnu_reconnect_m** command can use several couples login/password for the reconnection. In this previous case, the first parameter must be the session identifer.

2.2.1.4 Session management in VISHNU

After a successful call to the **vishnu_connect** command, a session is created. The session is required for calling any other commands. It avoids systematic authentification by userId and password. Only commands **vishnu_connect**, **vishnu_reconnect** and **vishnu_change_password** can be used outside a session by using userId and password. The **vishnu_list_history_cmd** command lists all the commands launched within a session.

To prevent unclosed sessions when the **vishnu_close** command is not used, the session is automatically closed on timeout or on disconnect (from the terminal).

2.2.1.4.1 Session close on timeout

In this mode, the session is automatically closed after an inactivity delay specified by the system or configured by the user using the **vishnu_configure_option** command.

2.2.1.4.2 Session close on disconnect

In this mode, the session is automatically closed when the shell terminal is closed. It is important to note that the system makes it impossible to close a session while commands are running. In this case, a session with automatic close on disconnect changes the close mode to automatic close on timeout.

2.2.1.5 Local user configuration management

2.2.1.5.1 Local user configuration creation

To access a UNIX account on a specific machine defined on VISHNU, the user must create a local user configuration by using the **vishnu_add_local_account** command. The **vishnu_list_machines** command gives information about the machines in which

a local user configuration can be created or where a local user configuration has already been created. The information required to create a new local user configuration are: the userId, the machineId, the login of the UNIX account on the specified machine, the absolute path to the user's private SSH key (used for file transfers) and the home directory path.

Warning: A unix login can only have one local account on a machine

The ssh public key of the machine named *"userId-machineId"* is returned and stored in the $HOME/.vishnu/localAccountPublicKey/ directory and must be added by the user in the ssh authorized key directory of the UNIX account. Doing this allows VISHNU to be directly connected on this UNIX account, running tasks as if it was the owner of the UNIX account.

With the command *"vishnu_set_ssh_key"* you can automatically add a local ssh public key to a remote account (in the authorized_keys file). This command also allows to remove a given key to that file. See the command reference section for the usage. **Note** that to enable rollback in case of an inappropriate removal, the command makes a backup of the authorized_keys file in a file named *"authorized_keys.bak"* before altering it.

2.2.1.5.2 Local user configuration update

All previous parameters used to create a local user configuration can be updated by using the **vishnu_update_local_account** command except for userId and machineId.

2.2.1.5.3 Local user configuration remove

A local user configuration can be removed by using the **vishnu_delete_local_account** command.

It is possible to display the local user configurations with the **vishnu_list_local_account** command. Other commands which are not cited above can be used to display information, such as the **vishnu_list_options** command, which displays all the options configured by the user, or the **vishnu_list_sessions** command, which displays information about the sessions.

2.2.1.6 Local user-authentication configuration management

2.2.1.6.1 Local user-authentication configuration creation

To be authenticated using a VISHNU user-authentication system different of the UMS database, the user must create a local user-authentication configuration by using the **vishnu_add_auth_account** command. The **vishnu_list_auth_systems** command gives information about the user-authentication systems in which a local user-authentication configuration can be created. The information required to create a new local user-authentication configuration are: the VISHNU identifier of the user-authentication system and the login of the user on this user-authentication system.

Warning: It is possible to define only one local user-authentication configuration on a specific user-authentication system for the same user.

2.2.1.6.2 Local user-authentication configuration update

Only the login used to create a local user-authentication configuration can be updated by using the **vishnu_update_auth_account** command.

2.2.1.6.3 Local user-authentication configuration remove

A local user-authentication configuration can be removed by using the **vishnu_delete_auth_account** command.

It is possible to display the local user-authentication configurations with the **vishnu_list_auth_accounts** command.

2.2.2 TMS package

2.2.2.1 Job submission

To submit a job, via VISHNU, to the batch scheduler of a specific machine, the user needs: an active VISHNU session, a local user configuration registered on VISHNU that corresponds to an existing UNIX account on the specified machine and a script that describes the job to submit. In the current implementation of VISHNU, it is possible to use the directives for several batch schedulers: TORQUE, LoadLeveler, SLURM, LSF and Grid Engine. In order to use the same script on different batch schedulers, a generic script with generic VISHNU directives is used. The **vishnu_submit_job** command allows a user to submit a job in the shell terminal. To obtain information on a job, the user can use the **vishnu_get_job_info** or **vishnu_list_jobs**, and for a job's progression status, **the vishnu_get_job_progression** command is used. The job's progression status is calculated according to the wall-clock time specified by the user during the job's submission.

2.2.2.1.1 VISHNU generic script

The key words of a VISHNU generic script start with the special character **#%**. For example, to specify a job's name, users have to use the following directive in their scripts: **#% vishnu_job_name**. The possible generic directives are:

- **#% vishnu_group**: allow to specify the group's name,
- **#% vishnu_working_dir**: allow to specify a job remote working dir,
- **#% vishnu_job_name**: allow to specify the job's name. Spaces are not accepted in job name,
- **#% vishnu_output**: allow to specify the path of the job's ouput file,
- **#% vishnu_error**: allow to specify the path of the file containing the problems that occured during the job's execution,
- **#% vishnu_wallclocklimit**: allow to specify the estimated time for the job's execution,
- **#% vishnu_cput**: allow to specify the job cpu limit time,
- **#% vishnu_nb_cpu**: allow to specify the number of cpus per node of the job,
- **#% vishnu_nbNodesAndCpuPerNode**: allow to specify the number of nodes and the cpu of each node. For exemple if you want to use 4 nodes and to use 3 cpus of each node, you must sepecify thes numbers by "4:3",
- **#% vishnu_memory**: allow to specify the memory size that the job requires,
- **#% vishnu_mailNotification** : allow to specify the notification type of the job. Valid type values are BEGIN, END, ERROR, and ALL (any state change),
- **#% vishnu_notify_user**: The name of user to receive email notification of state changes as defined by the option mailNotification. The default value is the submitting user,
- **#% vishnu_queue**: specifies the queue where the job will be submitted. It is possible to obtain a list of the batch scheduler's queues by using the **vishnu_list_queues** command.

It is important to note that the user can also add directives specific to a batch scheduler (TORQUE, LoadLeveler, SLURM, LSF or Grid Engine). Such specific directives must be added directly after the generic directives. Here is an example:

```
#!/bin/sh
#% vishnu_job_name=first_job
#% vishnu_queue=first_queue
#% vishnu_output=/path/to/jobOutput
#% vishnu_error=/path/to/jobError
#% vishnu_wallclocklimit=2:40:5
#This line is a comment
#The following lines are TORQUE specific section
#PBS -l ncpus=1
#PBS -l mem=50
```

```
        #The following lines are LOADLEVELER specific section
        #@ notify_user=user@mail
        #@ cpu_limit=2
        #The following lines are SLURM specific section
        #SBATCH -J myFristJob
        #SBATCH -o myJob-%j.out
        #SBATCH -e myJob-%j.err
        #SBATCH -t 01:02:20
        #SBATCH -p myFavoritePartition
        #The following lines are LSF specific section
        #BSUB -J myFristJob
        #BSUB -o myJob-%J.out
        #BSUB -e myJob-%J.err
        #BSUB -W 01:02
        #BSUB -q priority
#The following lines are Grid Engine specific section
#$ -N myFristJob
#$ -o myJob-$JOB_ID.out
#$ -e myJob-$JOB_ID.err
#$ -l s_rt=01:02:20
#$ -q myFavoriteQueue
```

2.2.2.2 JOB OUTPUT ENVIRONMENT VARIABLES

The VISHNU Job Manager set the following variables in the environment of the batch script.

- **VISHNU_BATCHJOB_ID**: Set the identifier assigned to the job by the batch system.
- **VISHNU_BATCHJOB_NAME**: Set the name of the job.
- **VISHNU_SUBMIT_MACHINE_NAME**: Set the name of the machine on which the job has been submitted.
- **VISHNU_BATCHJOB_NODEFILE**: Set the name of the file contain the list of nodes assigned to the job.
- **VISHNU_BATCHJOB_NUM_NODES**: Set the total number of nodes in the job's resource allocation.

2.2.2.3 Job Cancellation

To cancel a job, the **vishnu_cancel_job** command is used with the VISHNU identifer of the job to cancel. When the identifer of the job is *all*, all of the user's jobs are cancelled. An admin can also cancel all the jobs of all the users of VISHNU.

2.2.2.4 Job output files

VISHNU offers two commands, to be used in a shell terminal, to get the result output files for a job:

- **vishnu_get_job_output** or,
- **vishnu_get_completed_jobs_output**

The former gives the output files for a specific job while the latter gives the output files for all the completed jobs. It is important to note that all submitted jobs have two output files: one with the job's results, one (possibly empty) with the errors that occurred during the job's execution. The path of the job's output files is specified during the job's submission.

2.2.2.5 Configuration of ssh keys required for TMS

Submission, cancellation and getting of job output files are executed by TMS SeD launched via ssh under the account of user having issued the request. To execute these services correctly, the public ssh key of the account dedicated to TMS SeD must be added to authorized_keys ($HOME/.ssh/authorized_keys) file of the user. All keys protected by passphrase must be stored by a ssh agent to allow automatic authentication.

2.2.3 FMS package

2.2.3.1 Create and remove file or directories

The user can create (or remove) a regular file located in a remote host by using the command **vishnu_create_file** (or **vishnu_remove_file**
He can also create (or remove) directory located in a remote host by using the command **vishnu_create_dir** (or **vishnu_remove_dir**).

2.2.3.2 Get file information

Several services are available to get file information. The user can get the first (or last lines) of a given remote file by using the command **vishnu_head_of_file** (or **vishnu_tail_of_file**). He can also get the entire content of a remote file (or remote directory) with the command **vishnu_content_of_file** (or **vishnu_list__dir**).

2.2.3.3 Modify files properties

The commands **vishnu_ch_grp** and **vishnu_ch_mod** allow the user to change respectively the group and the access permissions of remote file.

2.2.3.4 Perform file transfer

The user can submit a file transfer in many ways:

- copy (or move) file between two hosts by the commands **vishnu_copy_file** (or **vishnu_move_file**). To use the copy/move commands from a client out of the DNS, one must use 'localhost' as a machine id for the local machine.
- copy (or move) file between two hosts in asynchronous way by the commands **vishnu_copy_async_file** (or **vishnu_move_async_file**)
- list or (cancel) file transfers by the commands **vishnu_list_file_transfers** (**vishnu_stop_file_transfers**)

2.2.4 IMS package

2.2.4.1 Export of commands

The user can export the commands made during a session in shell format. The file must have been created before the call, the file will contain a shell script that can be executed. For safety reasons, the connect and change password cannot be in the file. Thus, to execute the shell script, the connect command must be added after the first line of the file that declares the shell format.

2.2.4.2 Get system information

The get system information is to get information about a machine of the system. It can give information about the memory and/or diskspace available on the machine.

2.2.4.3 Get the metric

The user can get the current metrics on a machine or the history of the metrics on a machine to know the evolution of the metric. The user can choose to get the metrics of a specific type (parameters are 1, 2, 3 respectively free CPUload, free diskspace and free memory). Note, when getting the history of metrics, by default, only the first 1000 results are gotten. This is to avoid having to much results with a single command (the metric table may be filled very fastly).

2.2.5 Troubleshooting functions

- The *"There is no session in this terminal"* error can be solved by connecting to VISHNU using the **vishnu_connect** command.
- If a message similar to *"warning input stream error"* happens when calling vishnu_connect but the call is successful, try cleaning the .vishnu directory.
- The *"DIET initialization failed"* error, possibly caused by the number of omniorb connection limit, can be solved in the previous case by using the variable maxGIOPConnectionPerServer on the omniorb configuration file as follows : maxGIOPConnection-PerServer = 1000. It is also advisable to put the variable maxServerThreadPoolSize to 1000. The value 1000 is an example value.
- If the message *"locale::facet::_S_create_c_locale name not valid"* occurs after calling vishnu_connect command, you must export environment variable LANG as follows : export LANG=C.
- If the message *"The batch scheduler indicates an error [SGE ERROR: warning: username your job is not allowed to run in any queue]"* is returned after job submission on the batch scheduler SGE, you have to check the existence of a queue by using the command: **qconf -sql**. If there is no queue, you must add one by using the command **qconf -aq** .

2.2.6 FAQ

- If the JAVA API (using the WS or the JAVA jars directly fails), with an error such as undefined reference in a JNI call, please add the flags *-Wl,--add-stdcall-alias* at the compilation of the VISHNU target in the swigAPI/CMakeLists.txt.

Chapter 3

UMS Command reference

3.1 vishnu_connect

vishnu_connect — opens a session

Synopsis

`vishnu_connect` [`-h`] [`-p` *closePolicy*] [`-d` *sessionInactivityDelay*] [`-s` *substituteUserId*] [`-u` *userId*] [`-w` *password*]

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. The userId and password may be given through the options, or if no userId and password are specified, vishnu_connect will read them in the .netrc file located in the home of the user.

OPTIONS

`-h` ***`help`*** help about the command.

`-p` ***`closePolicy`*** is an option for closing session automatically. The value must be an integer. Predefined values are: 0 (DEFAULT), 1 (CLOSE_ON_TIMEOUT), 2 (CLOSE_ON_DISCONNECT).

`-d` ***`sessionInactivityDelay`*** is the maximum delay in seconds between two user requests when the CLOSE_ON_TIMEOUT policy is set.

`-s` ***`substituteUserId`*** is an admin option which allows an admin to open a session as if she was another user identified by her userId.

`-u` ***`userId`*** userId represents the VISHNU user identifier.

`-w` ***`password`*** password represents the password of the user.

ENVIRONMENT

`VISHNU_CLOSE_POLICY` The value of this environment variable represents the session close policy. Overridden by the -p option.

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is unknown or the password is wrong" [20]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

EXAMPLE

To connect the user user_1:

vishnu_connect -u user_1 -w pwd1

3.2 vishnu_connect_m

vishnu_connect_m — opens a session by trying multiple couples (userId, password) each in turn

Synopsis

`vishnu_connect_m [-h] [-p closePolicy] [-d sessionInactivityDelay] [-s substituteUserId] userId pwd`

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. It is possible to define multiple VISHNU couples (userId, password). In this case, vishnu_connect_m read them and tries each couple to make a connection.

OPTIONS

-h *help* help about the command.

-p *closePolicy* is an option for closing session automatically. The value must be an integer. Predefined values are: 0 (DEFAULT), 1 (CLOSE_ON_TIMEOUT), 2 (CLOSE_ON_DISCONNECT).

-d *sessionInactivityDelay* is the maximum delay in seconds between two user requests when the CLOSE_ON_TIMEOUT policy is set.

-s *substituteUserId* is an admin option which allows an admin to open a session as if she was another user identified by her userId.

ENVIRONMENT

VISHNU_CLOSE_POLICY The value of this environment variable represents the session close policy. Overridden by the -p option.

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is unknown or the password is wrong" [20]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

EXAMPLE

To connect the user user_1 or user_2 using a list of couples (userId, password) :

vishnu_connect_m user_1 pwd1 user_2 pwd2

3.3 vishnu_reconnect

vishnu_reconnect — reconnects to a session that is still active

Synopsis

`vishnu_reconnect` [`-h`] [`-u` *`userId`*] [`-w` *`password`*] *`sessionId`*

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The userId and password may be given through the options, or if no userId and password are specified, vishnu_reconnect will read them in the .netrc file located in the home of the user.

OPTIONS

`-h` ***`help`*** help about the command.

`-u` ***`userId`*** userId represents the VISHNU user identifier.

`-w` ***`password`*** password represents the password of the user.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The session Id is unknown" [30]

"The machine does not exist or it is locked" [36]

EXAMPLE

To reconnect the user user_1 to the session S01:

vishnu_reconnect -u user_1 S01

3.4 vishnu_reconnect_m

vishnu_reconnect_m — reconnects to a session that is still active by trying multiple couples (userId, password) each in turn

Synopsis

`vishnu_reconnect_m` [`-h`] *`sessionId userId pwd`*

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The vishnu_reconnect_m tries in turn multiple couples (userId, password).

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The session Id is unknown" [30]

"The machine does not exist or it is locked" [36]

EXAMPLE

To reconnect the user user_1 or user_2 to the session S01 using a list of couple userId, password :

vishnu_reconnect_m S01 user_1 pwd_1 user_2 pwd_2

Warning, the session id MUST be the first parameter in CLI

3.5 vishnu_close

vishnu_close — closes the session

Synopsis

`vishnu_close` [`-h`]

DESCRIPTION

This command closes the session that is currently active in the terminal. It will return an error if there are still some active requests that were been submitted by the user during the session (e.g., job submission or file transfers). After the session is closed, it cannot be re-opened.

OPTIONS

`-h` *`help`* help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"Commands are running" [31]

EXAMPLE

To close the current session:

vishnu_close

3.6 vishnu_change_password

vishnu_change_password — changes the password

Synopsis

`vishnu_change_password` [`-h`] *`userId`*

DESCRIPTION

This command is used to change the password. It can be done voluntarily or when the password is only temporary: for your first connection to VISHNU or after your password is reset by an administrator .

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"You can modify information. This account is read-only" [54]

EXAMPLE

To change the password of the user user_1:

vishnu_change_password user_1

3.7 vishnu_add_local_account

vishnu_add_local_account — adds a new local user configuration

Synopsis

`vishnu_add_local_account` [-h] *userId machineId acLogin sshKeyPath homeDirectory*

DESCRIPTION

A local user configuration must be added to allow you (identified by userId) to connect to machine (identified by machineId). This configuration must match an existing system account on that machine with a login that matches acLogin. The parameters sshKeyPath parameter (the absolute path to your private SSH key, used for file transfers) must be provided as well as the homeDirectory path .

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The machine is locked" [34]

"The machine does not exist or it is locked" [36]

"The local account already exists" [37]

"The system account login is already used by another vishnu user" [46]

EXAMPLE

To add a local account to the user user_1 on machine_1 with the login toto with the public key in .ssh/id_dsa.pub:
vishnu_add_local_account user_1 machine_1 toto /home/toto/.ssh/id_dsa.pub /home/toto

3.8 vishnu_update_local_account

vishnu_update_local_account — updates a local user configuration

Synopsis

`vishnu_update_local_account` [`-h`] [`-l` *acLogin*] [`-s` *sshKeyPath*] [`-d` *homeDirectory*] *userId machineId*

DESCRIPTION

The local user configuration can be updated. You can modify information of its local configuration such as acLogin, sshKeypath or homeDirectory.

OPTIONS

-h *help* help about the command.

-l *acLogin* acLogin represents the login of the user on the associated machine.

-s *sshKeyPath* sshKeyPath is the path of the ssh key of the user on the associated machine.

-d *homeDirectory* HomeDirectory is the path of the home directory of the user on the associated machine.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The system account login is already used by another vishnu user" [46]

EXAMPLE

To change the account login to toto2 for the user user_1 on machine_1:

vishnu_update_local_account user_1 machine_1 -l toto2

3.9 vishnu_delete_local_account

vishnu_delete_local_account — removes a local user configuration (for a given user on a given machine) from VISHNU

Synopsis

`vishnu_delete_local_account` [`-h`] *`userId machineId`*

DESCRIPTION

The local user configuration can be deleted from VISHNU. When a local user configuration is deleted, all the information about it is deleted from VISHNU.

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The local account is unknown" [38]

EXAMPLE

To delete the local account of the user user_1 on machine_1:

vishnu_delete_local_account user_1 machine_1

3.10 vishnu_list_local_accounts

vishnu_list_local_accounts — lists the local user configurations

Synopsis

`vishnu_list_local_accounts` [`-h`] [`-a`] [`-u` *`userId`*] [`-i` *`machineId`*]

DESCRIPTION

A local configuration is used to configure the access to a given system for a given user through VISHNU. It is related to an account on that system that is identified using its login. This command allows you to check all the local configurations related to your VISHNU account.

OPTIONS

`-h` ***`help`*** help about the command.

`-a` ***`adminListOption`*** is an admin option for listing all local configurations of all users .

`-u` ***`userId`*** is an admin option for listing the local configurations of a specific user .

`-i` ***`machineId`*** is an option for listing local user configurations on a specific machine.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

EXAMPLE

To list all the local accounts:

vishnu_list_local_accounts -a

3.11 vishnu_list_machines

vishnu_list_machines — lists the machines that are accessible through VISHNU

Synopsis

`vishnu_list_machines` [`-h`] [`-u` *`userId`*] [`-a`] [`-m` *`machineId`*]

DESCRIPTION

This command is used to display the machines that you can use for VISHNU services. The machines you can access through VISHNU are those that are configured in VISHNU by the VISHNU administrator, and that have been added to your personal VISHNU configuration using the vishnu_add_local_account command. The results contain, for each machine, a machine identifier that you can use as a parameter for other VISHNU commands.

OPTIONS

`-h` ***`help`*** help about the command.

`-u` ***`userId`*** is an admin option for listing machines in which a specific user has a local configuration.

`-a` ***`listAllMachine`*** is an option for listing all VISHNU machines.

`-m` ***`machineId`*** is an option for listing information about a specific machine.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

EXAMPLE

To list all the machines:

vishnu_list_machines -a

3.12 vishnu_list_history_cmd

vishnu_list_history_cmd — lists the commands

Synopsis

`vishnu_list_history_cmd` [`-h`] [`-a`] [`-u` *userId*] [`-i` *sessionId*] [`-s` *startDateOption*] [`-e` *endDateOption*]

DESCRIPTION

This command displays a history of the commands you ran. Several options can be used to specify which commands to list.

OPTIONS

`-h` ***help*** help about the command.

`-a` ***adminListOption*** is an admin option for listing all commands of all users.

`-u` ***userId*** is an admin option for listing commands launched by a specific user identified by his/her userId.

`-i` ***sessionId*** lists all commands launched within a specific session.

`-s` ***startDateOption*** allows the user to organize the commands listed by providing the start date (the UNIX timestamp of the start date is used).

`-e` ***endDateOption*** 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.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

EXAMPLE

To see the own history of commands of a user:

vishnu_list_history_cmd

3.13 vishnu_list_options

vishnu_list_options — lists the options of the user

Synopsis

`vishnu_list_options` [`-h`] [`-a`] [`-u` *`userId`*] [`-n` *`optionName`*]

DESCRIPTION

This command displays the options you configured.

OPTIONS

`-h` ***`help`*** help about the command.

`-a` ***`listAllDeftValue`*** is an option for listing all default option values defined by VISHNU administrator.

`-u` ***`userId`*** is an admin option for listing the options of a specific user.

`-n` ***`optionName`*** allows the user to get the value of a specific option identified by its name .

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The name of the user option is unknown" [41]

EXAMPLE

To list all the options of the user:

vishnu_list_options -a

3.14 vishnu_list_sessions

vishnu_list_sessions — lists all sessions of the user

Synopsis

`vishnu_list_sessions [-h] [-t status] [-p sessionClosePolicy] [-d sessionInactivityDelay] [-m machineId] [-a] [-u userId] [-i sessionId] [-s startDateOption] [-e endDateOption]`

DESCRIPTION

This command is used to display and filter the list of all your sessions. For each session, a session identifier is provided which you can use to reconnect to a given session using the vishnu_reconnect command. The session's status is either 0 (inactive) or 1 (active).

OPTIONS

-h ***help*** help about the command.

-t ***status*** specifies the status of the sessions which will be listed. The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (INACTIVE), 1 (ACTIVE).

-p ***sessionClosePolicy*** specifies the closure mode of the sessions which will be listed (CLOSE_ON_TIMEOUT or CLOSE_ON_D The value must be an integer. Predefined values are: 0 (DEFAULT), 1 (CLOSE_ON_TIMEOUT), 2 (CLOSE_ON_DISCONNECT

-d ***sessionInactivityDelay*** specifies the inactivity delay in seconds of the sessions which will be listed.

-m ***machineId*** allows the user to list sessions opened on a specific client's machine by using its name.

-a ***adminListOption*** is an admin option for listing all sessions of all users.

-u ***userId*** is an admin option for listing sessions opened by a specific user.

-i ***sessionId*** allows the user to list all commands launched within a specific session.

-s ***startDateOption*** allows the user to organize the commands listed by providing the start date (the UNIX timestamp of the start date is used).

-e ***endDateOption*** 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.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The closure policy is unknown" [42]

EXAMPLE

To list all opened the sessions of the user user_1:

vishnu_list_session -u user_1

3.15 vishnu_configure_option

vishnu_configure_option — configures an option of the user

Synopsis

`vishnu_configure_option` [`-h`] *`optionName value`*

DESCRIPTION

Options in VISHNU corresponds to the parameters of some VISHNU commands (e.g., the close policy for vishnu_connect) that can be preset in the user configuration stored by the VISHNU system. This command is used to set the value of an option for the current user (the user who opened the session).

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The name of the user option is unknown" [41]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

"The value of the transfer command is incorrect" [44]

EXAMPLE

To set the value of the option VISHNU_TIMEOUT to the value 69:

vishnu_configure_option VISHNU_TIMEOUT 69

3.16 vishnu_current_session_id

vishnu_current_session_id — display the session id

Synopsis

`vishnu_current_session_id [-h]`

DESCRIPTION

This command allows to get current session identifier.

OPTIONS

`-h help` help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Undefined error" [16]

EXAMPLE

To get the current session identifier:

vishnu_current_session_id

3.17 vishnu_list_auth_systems

vishnu_list_auth_systems — lists VISHNU user-authentification systems

Synopsis

`vishnu_list_auth_systems` [`-h`] [`-a`] [`-f`] [`-u` *`userId`*] [`-i` *`authSystemId`*]

DESCRIPTION

This command allows to display all user-authentication systems. By default, the user-authentication systems where the user has a local user-authentication config are listed.

OPTIONS

`-h` ***`help`*** help about the command.

`-a` ***`listAllAuthSystems`*** is an option for listing all VISHNU user-authentication systems.

`-f` ***`listFullInfo`*** is an admin option for listing full VISHNU user-authentication systems information such as all concerned only the administrator: authLogin, authPassword and userPasswordEncryption .

`-u` ***`userId`*** is an admin option for listing all user-authentication systems in which a specific user has local user-authentication configs.

`-i` ***`authSystemId`*** is an option for listing a specific user-authentication system.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The type of the user-authentication system is unknown" [47]

EXAMPLE

To list all user-authentication systems:

vishnu_list_auth_systems

3.18 vishnu_add_auth_account

vishnu_add_auth_account — adds a new local user-authentication configuration

Synopsis

`vishnu_add_auth_account` [`-h`] [`-u` *`userId`*] *`authSystemId`* *`acLogin`*

DESCRIPTION

This command allows to add a local user-authentication configuration in VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

OPTIONS

`-h` ***`help`*** help about the command.

`-u` ***`userId`*** represents an admin option to add a user-authentication account of a specific user.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The system account login is already used by another vishnu user" [46]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account already exists" [51]

EXAMPLE

To add the local user-authentication configuration whose account is toto on the user-authentication system identified by AUTHENLDAP_1:

vishnu_add_auth_account AUTHENLDAP_1 toto

3.19 vishnu_update_auth_account

vishnu_update_auth_account — updates a local user-authentication configuration

Synopsis

`vishnu_update_auth_account` [`-h`] [`-u` *`userId`*] [`-l` *`acLogin`*] *`authSystemId`*

DESCRIPTION

This command allows to update a local user-authentication configuration in VISHNU. Only the local user's login in the user-authentication system can be updated.

OPTIONS

`-h` ***`help`*** help about the command.

`-u` ***`userId`*** represents an admin option to add a user-authentication account of a specific user.

`-l` ***`acLogin`*** is an option to change the login of the user on the associated user-authentication system.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

EXAMPLE

To change the uid of a local user-authentication configuration to aramis whose user-authentication system is AUTHENLDAP_1:
vishnu_update_auth_account -l aramis AUTHENLDAP_1

3.20 vishnu_delete_auth_account

vishnu_delete_auth_account — removes a local user-authentication configuration from VISHNU

Synopsis

`vishnu_delete_auth_account` [`-h`] [`-u` *`userId`*] *`authSystemId`*

DESCRIPTION

This command allows to remove a local user-authentication configuration from VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

OPTIONS

-h *help* help about the command.

-u *userId* is an admin option which represents the VISHNU identifier of the user whose local user-authentication configuration will be deleted.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

EXAMPLE

To remove a local user-authentication configuration whose user-authentication system is AUTHENLDAP_1:

vishnu_delete_auth_account AUTHENLDAP_1

3.21 vishnu_list_auth_accounts

vishnu_list_auth_accounts — lists local user-authentication configurations

Synopsis

`vishnu_list_auth_accounts [-h] [-a] [-u userId] [-i authSystemId]`

DESCRIPTION

This command allows to list local user-authentication configurations of VISHNU.

OPTIONS

`-h help` help about the command.

`-a listAll` is an admin option for listing all local user-authentication configurations of VISHNU.

`-u userId` is an admin option for listing all local user-authentication configurations of a specific user identified by his/her userId.

`-i authSystemId` is an option for listing local user-configuration of a specific user-authentication system.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

EXAMPLE

To list local user-authentication configurations:

vishnu_list_auth_accounts

3.22 vishnu_set_ssh_key

vishnu_set_ssh_key — Allows to add or remove a given key to/from the authorized keys of a user's remote account.

Synopsis

`vishnu_set_ssh_key [-h] [-a] [-r] rlogin sshPubKey`

DESCRIPTION

This command allows to add or remove a given key to/from the authorized keys of a user's remote account

OPTIONS

`-h help` help about the command.

`-a addOption` option for adding a key.

`-r removeOption` option for removing a key.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"an option or a parameter provided is invalid for this service" [10]

"Undefined error" [16]

"The name of the user option is unknown" [41]

EXAMPLE

To add the public key located localy at $HOME/.ssh/rsa_id.pub to the authorized_keys of the user tartampion on the machine named slurm.intranet.lan :

vishnu_set_ssh_key -a $HOME/.ssh/rsa_id.pub tartampion@slurm.intranet.lan

To remote the key added previously:

vishnu_set_ssh_key -r $HOME/.ssh/rsa_id.pub tartampion@slurm.intranet.lan

Chapter 4

TMS Command reference

4.1 vishnu_submit_job

vishnu_submit_job — Allows to submit a job consisting in running a given script on a machine.

Synopsis

vishnu_submit_job [-h] [-n *name*] [-q *queue*] [-t *wallTime*] [-m *memory*] [-P *nbCpu*] [-N *nbNodesAndCpuPerNode*] [-o *outputPath*] [-e *errorPath*] [-M *mailNotification*] [-u *mailNotifyUser*] [-g *group*] [-D *workingDir*] [-T *cpuTime*] [-Q] [-L *loadType*] [-F *fileParams*] [-V *textParams*] [-w *workId*] *machineId scriptFilePath*

DESCRIPTION

This command is used to submit a job to the specific batch scheduler associated to a machine. It allows describing a job in a script, using either the batch scheduler's directives or VISHNU's generic directives for all batch schedulers. If the machine identifier is equal to autom, the job will be automatically submitted on a best machine (for now three criterions are used: minimum number of waiting jobs, minimum number of running jobs and the total number of jobs) through the use of a script (scriptFilePath) which must be generic script using VISHNU's generic directives for all batch schedulers

OPTIONS

-h *help* help about the command.

-n *name* Assigns a job name. The default is the path of job.

-q *queue* Assigns the queue or class of the job.

-t *wallTime* The maximum wall-clock time during which the job can run (in seconds).

-m *memory* Is the memory size that the job requires (in MegaBytes).

-P *nbCpu* The number of cpu per node that the job requires.

-N *nbNodesAndCpuPerNode* The number of nodes and processors per node (in the format nbNodes:nbCpuPerNode). For example if you want to use 4 nodes with 3 cpus for each node, you must specify these numbers by "4:3".

-o *outputPath* Assigns the path and file for job output.

-e *errorPath* Assigns the path and file for job error.

-M *mailNotification* Assigns the notification type of the job. Valid type values are BEGIN, END, ERROR, and ALL (any state change).

-u *mailNotifyUser* The name of user to receive email notification of state changes as defined by the option mailNotification. The default value is the submitting user.

-g *group* Assigns a job group name..

-D *workingDir* Assigns a job remote working dir.

-T *cpuTime* Assigns a job cpu limit time (in seconds or in the format [[HH:]MM:]SS).

-Q *selectQueueAutom* allows to select automatically a queue which has the number of nodes requested by the user.

-L *loadType* The criterion to automatically submit a job (for now three criterions are used: minimum number of waiting jobs, minimum number of running jobs and the total number of jobs). This option is used only if the machine identifier is equal to autom (this keyword is used to submit automatically a job). The value must be an integer. Predefined values are: 0 (USE_NB_WAITING_JOBS), 1 (USE_NB_JOBS), 2 (USE_NB_RUNNING_JOBS).

-F *fileParams* Sets a list of local files as parameters for the script. These files will be uploaded onto the server before computing the script. At the execution time, each path can be accessed through an environment variable. E.g. --fileParams "PFILE1=/path/to/file1 PFILE2=/path/to/file2" or -F "PFILE1=/path/to/file1 PFILE2=/path/to/file2". At the execution, the environment variable $PFILE1, for example, will point to the uploaded copy of the file /path/to/file1. The option --fileParam or -f allows to set each file individually. NOTE: The name of parameter as well as those of the corresponding environment are case-insensitive. .

-V *textParams* Sets a list of parameters for the script so at the execution, each of parameter can be used as an environment variable. E.g. --textParams "PARAM1=value1 PARAM2=value2" or -F "PARAM1=value1 PARAM2=value2", will set the environment variables named PARAM1 and PARAM2 with value1 and value2, respectively. The option --textParam or -v allows to set each file individually. NOTE: The name of parameter as well as those of the corresponding environment are case-insensitive..

-w *workId* Sets the identifier of the Work to which the job is related.

JOB OUTPUT ENVIRONMENT VARIABLES

The VISHNU Job Manager set the following variables in the environment of the batch script.

VISHNU_BATCHJOB_ID Set the identifier assigned to the job by the batch system.

VISHNU_BATCHJOB_NAME Set the name of the job.

VISHNU_SUBMIT_MACHINE_NAME Set the name of the machine on which the job has been submitted.

VISHNU_BATCHJOB_NODEFILE Set the name of the file contain the list of nodes assigned to the job.

VISHNU_BATCHJOB_NUM_NODES Set the total number of nodes in the job's resource allocation.

VISHNU_OUTPUT_DIR Define a generic directory where files generated through a script would be stored in order to be get out easily using the related VISHNU commands (vishnu_get_output or vishnu_get_completed_jobs_output). Note that the real path of the directory is a set at the execution time with a directory automatically created within the job's working directory by the system.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"Error invalid parameters" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To submit on machine_1 the script toto:

vishnu_submit_job machine_1 toto

To submit automatically the script toto on the best machine (by the default the machine which has the minimum number of waiting jobs is selected):

vishnu_submit_job autom toto

To submit automatically the script toto on the best machine by using a machine which has the minimum total number of jobs:

vishnu_submit_job autom toto -L 1

4.2 vishnu_get_job_info

vishnu_get_job_info — gets information on a job from its id

Synopsis

`vishnu_get_job_info` [`-h`] *`machineId jobId`*

DESCRIPTION

This command allows getting information about a specific job. It can return the job's status, for example.

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To get the info on the job J_1 on machine_1:
vishnu_get_job_info machine_1 J_1

4.3 vishnu_get_job_progress

vishnu_get_job_progress — gets the progression status of jobs

Synopsis

`vishnu_get_job_progress` [`-h`] [`-i` *jobId*] [`-u` *jobOwner*] *machineId*

DESCRIPTION

This command allows getting the progression status of a job based on the wall-clock time specified.

OPTIONS

-h *help* help about the command.

-i *jobId* Specifies the id of the job whose progression the user wants to see..

-u *jobOwner* Specifies the owner of the job..

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

EXAMPLE

To get the progress of the job J_1 on machine_1:
vishnu_get_job_progress machine_1 -i J_1

4.4 vishnu_list_queues

vishnu_list_queues — gets queues information

Synopsis

`vishnu_list_queues` [`-h`] [`-q` *`queueName`*] *`machineId`*

DESCRIPTION

This command displays the status of the queues of a specific machine's batch scheduler.

OPTIONS

-h *help* help about the command.

-q *queueName* if it is given, listQueues gives information only of this queue.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To list the queues available on machine_1:

vishnu_list_queues machine_1

4.5 vishnu_list_jobs

vishnu_list_jobs — gets a list of all submitted jobs on a machine.

Synopsis

`vishnu_list_jobs [-h] [-i jobId] [-P nbCpu] [-d fromSubmitDate] [-D toSubmitDate] [-u owner] [-s status] [-p priority] [-q queue] [-S multipleStatus] [-b] [-w workId] machineId`

DESCRIPTION

This command allows displaying the jobs submitted on a specific machine's batch scheduler. If machine identifier is equal to all, submitted jobs on all machines are listed

OPTIONS

-h *help* help about the command.

-i *jobId* lists the job with the specified id.

-P *nbCpu* lists the jobs with the specified number of CPUs per node.

-d *fromSubmitDate* lists the jobs submitted after the specified date (UNIX timestamp).

-D *toSubmitDate* lists jobs submitted before the specified date (UNIX timestamp).

-u *owner* lists the jobs submitted by the specified owner.

-s *status* lists the jobs with the specified status. The value must be an integer. Predefined values are: -1 (UNDEFINED), 1 (SUBMITTED), 2 (QUEUED), 3 (WAITING), 4 (RUNNING), 5 (TERMINATED), 6 (CANCELLED), 7 (ALREADY_DOWNLO

-p *priority* lists the jobs with the specified priority. The value must be an integer. Predefined values are: -1 (UNDEFINED), 1 (VERY_LOW), 2 (LOW), 3 (NORMAL), 4 (HIGH), 5 (VERY_HIGH).

-q *queue* the jobs with the specified queue name.

-S *multipleStatus* lists the jobs with the specified status (combination of multiple status). Its format contains the first letter or the value (interger) of each chosen status. For exemple to list the cancelled and terminated jobs, you use the format CT or 65.

-b *batchJob* allows to select all jobs submitted through the underlying batch scheduler (jobs submitted through vishnu and out of vishnu).

-w *workId* Allows to gather information about jobs related to a given Work..

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To list the jobs on machine_1:

vishnu_list_jobs machine_1

To list submitted jobs on all machines:

vishnu_list_jobs all

4.6 vishnu_get_job_output

vishnu_get_job_output — gets standard output and error output files of a job given its id

Synopsis

`vishnu_get_job_output` [`-h`] [`-o` *`outDir`*] *`machineId`* *`jobId`*

DESCRIPTION

This command allows getting a job's output files.

OPTIONS

`-h` ***`help`*** help about the command.

`-o` ***`outDir`*** The output directory where the files will be stored (default is current directory).

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

"The job is not terminated" [107]

"The job is already downloaded" [108]

EXAMPLE

To get the output of the job J_1 on machine_1:

vishnu_get_job_output machine_1 J_1

4.7 vishnu_get_completed_jobs_output

vishnu_get_completed_jobs_output — gets standard output and error output files of completed jobs (applies only once for each job)

Synopsis

`vishnu_get_completed_jobs_output` [`-h`] [`-o` *`outDir`*] *`machineId`*

DESCRIPTION

This command allows getting the output files of all the completed jobs.

OPTIONS

-h *help* help about the command.

-o *outDir* Specifies the output directory where the files will be stored (by default, the current directory).

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To get the completed job on machine_1:

vishnu_get_completed_jobs_output machine_1

4.8 vishnu_cancel_job

vishnu_cancel_job — Allows to cancel a job submitted on a given machine.

Synopsis

`vishnu_cancel_job [-h] machineId jobId`

DESCRIPTION

cancels a job from its id. If job id is equal to all, all submitted jobs by all users will be cancelled if the user is an administrator, and only jobs submitted by the user will be cancelled if the user is not an administrator.

OPTIONS

`-h` *`help`* help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

"The job is already terminated" [105]

"The job is already canceled" [106]

EXAMPLE

To cancel the submission of the job J_1 on machine_1:

vishnu_cancel_job machine_1 J_1

To cancel cancel all submitted jobs on machine_1

vishnu_cancel_job machine_1 all

4.9 vishnu_add_work

vishnu_add_work — Allows to create a work consisting in running jobs.

Synopsis

`vishnu_add_work` [`-h`]

DESCRIPTION

This command adds a work in vishnu

OPTIONS

`-h` ***`help`*** help about the command.

JOB OUTPUT ENVIRONMENT VARIABLES

The VISHNU Job Manager set the following variables in the environment of the batch script.

`VISHNU_BATCHJOB_ID` Set the identifier assigned to the job by the batch system.

`VISHNU_BATCHJOB_NAME` Set the name of the job.

`VISHNU_SUBMIT_MACHINE_NAME` Set the name of the machine on which the job has been submitted.

`VISHNU_BATCHJOB_NODEFILE` Set the name of the file contain the list of nodes assigned to the job.

`VISHNU_BATCHJOB_NUM_NODES` Set the total number of nodes in the job's resource allocation.

`VISHNU_OUTPUT_DIR` Define a generic directory where files generated through a script would be stored in order to be get out easily using the related VISHNU commands (vishnu_get_output or vishnu_get_completed_jobs_output). Note that the real path of the directory is a set at the execution time with a directory automatically created within the job's working directory by the system.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"Error invalid parameters" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

EXAMPLE

To add a work called toto:

vishnu_add_work toto

Chapter 5

FMS Command reference

5.1 vishnu_create_file

vishnu_create_file — creates files on remote machines.

Synopsis

`vishnu_create_file` [`-h`] *`path`*

DESCRIPTION

Creates an empty file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To create the file toto on machine_1:

vishnu_create_file machine_1:/tmp/toto

5.2 vishnu_create_dir

vishnu_create_dir — creates directories on remote machines.

Synopsis

`vishnu_create_dir` [`-h`] [`-p`] *`path`*

DESCRIPTION

Creates an new directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

OPTIONS

`-h` ***`help`*** help about the command.

`-p` ***`isRecursive`*** It specifies when the create command is recursive (create parent directory also) or not.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To create the repository toto on machine_1:

vishnu_create_dir machine_1:/tmp/toto

5.3 vishnu_remove_file

vishnu_remove_file — removes files from remote hosts.

Synopsis

`vishnu_remove_file` [`-h`] [`-r`] *`path`*

DESCRIPTION

Deletes a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

OPTIONS

`-h` ***`help`*** help about the command.

`-r` ***`isRecursive`*** It specifies when the remove command is recursive (case of directory) or not.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To remove the file toto on machine_1:

vishnu_remove_file machine_1:/tmp/toto

5.4 vishnu_remove_dir

vishnu_remove_dir — removes directories (and subdirectories) from remote machines.

Synopsis

`vishnu_remove_dir` [`-h`] *`path`*

DESCRIPTION

Deletes a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To remove the repository toto on machine_1:

vishnu_remove_dir machine_1:/tmp/toto

5.5 vishnu_ch_grp

vishnu_ch_grp — changes group owner of remote files/directories.

Synopsis

`vishnu_ch_grp` [`-h`] *`group path`*

DESCRIPTION

Changes the group attribute of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To change the group owner to test of the file toto on machine_1:

vishnu_ch_owner test machine_1:/tmp/toto

5.6 vishnu_ch_mod

vishnu_ch_mod — changes access rights of remote files/directories.

Synopsis

`vishnu_ch_mod` [-h] *mode path*

DESCRIPTION

Changes the permissions of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command). The mode parameter is the same value as for the unix chmod command.

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To change the access rights on file toto on machine_1 to all rights:

vishnu_ch_mod 777 machine_1:/tmp/toto

5.7 vishnu_head_of_file

vishnu_head_of_file — displays a few first lines of files located on remote machines.

Synopsis

`vishnu_head_of_file` [-h] [-n *nline*] *path*

DESCRIPTION

Displays the first lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

OPTIONS

-h *help* help about the command.

-n *nline* the number of line to get.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To visualize the 3 first line on the file toto on machine_1:

vishnu_head_of_file -n 3 machine_1:/tmp/toto

5.8 vishnu_tail_of_file

vishnu_tail_of_file — displays a few last lines of files located on remote machines

Synopsis

`vishnu_tail_of_file` [-h] [-n *nline*] *path*

DESCRIPTION

Displays the last lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

OPTIONS

-h *help* help about the command.

-n *nline* the number of line to get.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To vizualize the 3 last lines of the file toto on machine_1:

vishnu_tail_of_file -n 3 machine_1:/tmp/toto

5.9 vishnu_content_of_file

vishnu_content_of_file — displays content of files located on remote machines

Synopsis

`vishnu_content_of_file` [-h] *path*

DESCRIPTION

Displays the content of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To view the content of the file toto on machine_1:

vishnu_content_of_file machine_1:/tmp/toto

5.10 vishnu_list_dir

vishnu_list_dir — displays the content of a remote directory

Synopsis

`vishnu_list_dir [-h] [-l] [-a] path`

DESCRIPTION

Displays the content of a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

OPTIONS

-h *help* help about the command.

-l *longFormat* It specifies the long display format (all available file informations).

-a *allFiles* Allows to display all files including hidden files.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To display the content of the toto repository on machine_1:

vishnu_list_dir machine_1:/tmp/toto

5.11 vishnu_copy_file

vishnu_copy_file — executes a synchronous copy of file.

Synopsis

`vishnu_copy_file` [`-h`] [`-r`] [`-t` *trCommand*] *src dest*

DESCRIPTION

Copy a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command.

OPTIONS

-h *help* help about the command.

-r *isRecursive* It specifies when the copy is recursive (case of directory) or not.

-t *trCommand* the command to use to perform file transfer. The value must be an integer. Predefined values are: 0 (SCP), 1 (RSYNC), 2 (UNDEFINED).

ENVIRONMENT

VISHNU_TRANSFER_CMD It specifies the command to use for all file transfers by default. It takes its values in the set {SCP,RSYNC}.. Overridden by the -t option.

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To copy the file toto from machine_1 to machine_2 home directory:

vishnu_copy_file machine_1:/tmp/toto machine_2:/home/vishnu/

5.12 vishnu_copy_async_file

vishnu_copy_async_file — executes an asynchronous copy of file.

Synopsis

`vishnu_copy_async_file [-h] [-r] [-t trCommand] src dest`

DESCRIPTION

Initiates a copy of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

OPTIONS

-h *help* help about the command.

-r *isRecursive* It specifies when the copy is recursive (case of directory) or not.

-t *trCommand* the command to use to perform file transfer. The value must be an integer. Predefined values are: 0 (SCP), 1 (RSYNC), 2 (UNDEFINED).

ENVIRONMENT

VISHNU_TRANSFER_CMD It specifies the command to use for all file transfers by default. It takes its values in the set {SCP,RSYNC}.. Overridden by the -t option.

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To copy using an asynchrone method the file toto from machine_1 to machine_2 home directory:

vishnu_copy_async_file machine_1:/tmp/toto machine_2:/home/vishnu/

5.13 vishnu_move_file

vishnu_move_file — executes a synchronous move of file.

Synopsis

`vishnu_move_file [-h] [-r] [-t trCommand] src dest`

DESCRIPTION

Move a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command.

OPTIONS

-h *help* help about the command.

-r *isRecursive* It specifies when the copy is recursive (case of directory) or not.

-t *trCommand* the command to use to perform file transfer. The value must be an integer. Predefined values are: 0 (SCP), 1 (RSYNC), 2 (UNDEFINED).

ENVIRONMENT

VISHNU_TRANSFER_CMD It specifies the command to use for all file transfers by default. It takes its values in the set {SCP,RSYNC}.. Overridden by the -t option.

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To move the file toto from machine_1 to machine_2 home directory:

vishnu_move_file machine_1:/tmp/toto machine_2:/home/vishnu/

5.14 vishnu_move_async_file

vishnu_move_async_file — executes an asynchronous move of file.

Synopsis

`vishnu_move_async_file [-h] [-r] [-t trCommand] src dest`

DESCRIPTION

Initiates a move of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

OPTIONS

-h *help* help about the command.

-r *isRecursive* It specifies when the copy is recursive (case of directory) or not.

-t *trCommand* the command to use to perform file transfer. The value must be an integer. Predefined values are: 0 (SCP), 1 (RSYNC), 2 (UNDEFINED).

ENVIRONMENT

VISHNU_TRANSFER_CMD It specifies the command to use for all file transfers by default. It takes its values in the set {SCP,RSYNC}.. Overridden by the -t option.

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To move using an asynchronous method the file toto from machine_1 to machine_2 home directory:

vishnu_move_async_file machine_1:/tmp/toto machine_2:/home/vishnu/

5.15 vishnu_stop_file_transfer

vishnu_stop_file_transfer — stops an execution of a set of file transfers.

Synopsis

`vishnu_stop_file_transfer` [`-h`] [`-i` *`transferId`*] [`-m` *`fromMachineId`*] [`-u` *`userId`*]

DESCRIPTION

Cancels a file or directory transfer that has been initiated using a vishnu asynchronous copy or move file command. The command listFileTransfers can be used to check the status of the transfer after it has been cancelled, using the transfer id as the identifier.

OPTIONS

-h *help* help about the command.

-i *transferId* a given transfer id.

-m *fromMachineId* the machine that is the source or destination of the file transfer.

-u *userId* allows an admin to stop file transfers of a specific user.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To stop the file transfers on machine_1:

vishnu_stop_file_transfer -m machine_1

5.16 vishnu_list_file_transfers

vishnu_list_file_transfers — displays the history of all file transfers submitted by User.

Synopsis

`vishnu_list_file_transfers [-h] [-t transferId] [-m fromMachineId] [-u userId] [-s status]`

DESCRIPTION

Get the list of all file or directory transfers that have been initiated using a vishnu synchronous or asynchronous copy or move file command.

OPTIONS

-h ***help*** help about the command.

-t ***transferId*** a given transfer id.

-m ***fromMachineId*** the machine that is the source of the file transfer.

-u ***userId*** allows the admin to list file transfers initiated by a specific user.

-s ***status*** the file transfer status. The value must be an integer. Predefined values are: 0 (INPROGRESS), 1 (COMPLETED), 2 (CANCELLED), 3 (FAILED), 4 (UNDEFINED).

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To list the file transfers on machine_1 for the user user_1:

vishnu_list_file_transfers -m machine_1 -u user_1

5.17 vishnu_get_file_info

vishnu_get_file_info — displays the information of files.

Synopsis

`vishnu_get_file_info` [`-h`] *`path`*

DESCRIPTION

Get the details of a remote file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

OPTIONS

`-h` ***`help`*** help about the command.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"Vishnu not available (Database error)" [2]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

EXAMPLE

To get the data concerning the file toto on machine_1:

vishnu_get_file_info machine_1:/tmp/toto

Chapter 6

IMS Command reference

6.1 vishnu_export_commands

vishnu_export_commands — exports all the commands made by a user during a session

Synopsis

`vishnu_export_commands` [`-h`] [`-t` *`exportType`*] *`oldSessionId`*

DESCRIPTION

Exports all the VISHNU commands submitted during a completed session. This session must be in closed state. The output of this command is a file containing a shell script. For safety reasons, the commands having a password for parameter are not exported (for example the vishnu_connect and vishnu_change_password commands). This means the shell script must be run after opening a session manually or by adding the vishnu_connect command to the script. The access to other user's sessions is only permitted to administrators.

OPTIONS

`-h` ***`help`*** help about the command.

`-t` ***`exportType`*** The type to export. The value must be an integer. Predefined values are: 0 (UNDEFINED), 1 (SHELL).

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

EXAMPLE

To export the commands made during the session with id S01 in the file toto:

vishnu_export_commands S01 /tmp/toto

6.2 vishnu_get_metric_current_value

vishnu_get_metric_current_value — displays the current values of system metrics

Synopsis

`vishnu_get_metric_current_value` [`-h`] [`-t` *`metricType`*] *`machineId`*

DESCRIPTION

Displays the current values of the monitored metrics on the system identified by the machineId argument : cpuload, free diskspace and free memory. The units of displayed values are percentages for cpuload and Megabytes (Mb) for diskspace and memory. The provided values are always standard integers (no float values). Please note that retrieving these values uses some valuable system ressources and should not occur too frequently to avoid an impact on system performance.

OPTIONS

`-h` ***`help`*** help about the command.

`-t` ***`metricType`*** The type of the metric. The value must be an integer. Predefined values are: 0 (UNDEFINED), 1 (CPUUSE), 2 (FREEDISKSPACE), 3 (FREEMEMORY).

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

EXAMPLE

To get the current values of the metrics on machine_1:

vishnu_get_metric_current_value machine_1

6.3 vishnu_get_metric_history

vishnu_get_metric_history — displays the history of values of a system metric

Synopsis

`vishnu_get_metric_history` [`-h`] [`-s` *startTime*] [`-e` *endTime*] [`-t` *type*] *machineId*

DESCRIPTION

Displays the chronological list of values of the metrics on the system identified by the machineId argument. Using the options it is possible to specify a type of metric and the starting and ending dates of the desired monitoring period. Note that some data will be available only if the required VISHNU agent (IMS server) has been running locally on the machine during the specified period.

OPTIONS

`-h` ***`help`*** help about the command.

`-s` ***`startTime`*** The start time to get the history.

`-e` ***`endTime`*** The end time to get the history.

`-t` ***`type`*** The type of metric searched. The value must be an integer. Predefined values are: 0 (UNDEFINED), 1 (CPUUSE), 2 (FREEDISKSPACE), 3 (FREEMEMORY).

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

EXAMPLE

To get the history of the metrics on machine_1:

vishnu_get_metric_history machine_1

6.4 vishnu_get_update_frequency

vishnu_get_update_frequency — gets the update frequency of the IMS database

Synopsis

`vishnu_get_update_frequency` [`-h`]

DESCRIPTION

This function allows a user to get the update frequency, to know how often the state of the machines is automatically polled to get historical data.

OPTIONS

-h *help* help about the command.

ENVIRONMENT

VISHNU_CONFIG_FILE Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

EXAMPLE

To get the update frequency:

vishnu_get_update_frequency

6.5 vishnu_get_system_info

vishnu_get_system_info — To get the system info on a machine

Synopsis

`vishnu_get_system_info [-h] [-m machineId]`

DESCRIPTION

This function allows a user to get system information about a machine. A system information describes a machine. The option is the machine id (if no machine id, the information for all the machines are given)

OPTIONS

`-h help` help about the command.

`-m machineId` The machine id.

ENVIRONMENT

`VISHNU_CONFIG_FILE` Contains the path to the local configuration file for VISHNU.

DIAGNOSTICS

The following diagnostics may be issued on stderr and the command will return the code provided within brackets:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"There is no open session in this terminal" [13]

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

EXAMPLE

To get all the system info for all the machine:

vishnu_get_system_info

Chapter 7

UMS C++ API Reference

7.1 connect

connect — opens a session

Synopsis

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

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. The userId and password may be given through the options, or if no userId and password are specified, vishnu_connect will read them in the .netrc file located in the home of the user.

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier. If userId and password are empty, vishnu connect will read them in the .netrc file located in the home of the user.

password Input argument. Password represents the password of the user. If userId and password are empty, vishnu connect will read them in the .netrc file located in the home of the user.

session Output argument. The session object that contains the created session details.

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The user is unknown or the password is wrong" [20]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

7.2 connect

connect — opens a session by trying multiple couples (userId, password) each in turn

Synopsis

int **vishnu::connect**(const ListUsers& listUsers, Session& session, const ConnectOptions& options = ConnectOptions());

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. It is possible to define multiple VISHNU couples (userId, password). In this case, vishnu_connect_m read them and tries each couple to make a connection.

ARGUMENTS

listUsers Input argument. The list containing one or more couple (userId and password) potentially usable for connection.

session Output argument. The session object that contains the created session details.

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The user is unknown or the password is wrong" [20]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

7.3 reconnect

reconnect — reconnects to a session that is still active

Synopsis

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

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The userId and password may be given through the options, or if no userId and password are specified, vishnu_reconnect will read them in the .netrc file located in the home of the user.

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier. If userId and password are empty, vishnu reconnect will read them in the .netrc file located in the home of the user.

password Input argument. Password represents the password of the user. If userId and password are empty, vishnu reconnect will read them in the .netrc file located in the home of the user.

sessionId Input argument. SessionId is the identifier of the session defined in the database.

session Output argument. The session object containing session information.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The session Id is unknown" [30]

"The machine does not exist or it is locked" [36]

7.4 reconnect

reconnect — reconnects to a session that is still active by trying multiple couples (userId, password) each in turn

Synopsis

int **vishnu::reconnect**(const ListUsers& listUsers, const string& sessionId, Session& session);

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The vishnu_reconnect_m tries in turn multiple couples (userId, password).

ARGUMENTS

listUsers Input argument. The list containing one or more couple (userId and password) potentially usable for connection.

sessionId Input argument. SessionId is the identifier of the session defined in the database.

session Output argument. The session object containing session information.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The session Id is unknown" [30]

"The machine does not exist or it is locked" [36]

7.5 close

close — closes the session

Synopsis

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

DESCRIPTION

This command closes the session that is currently active in the terminal. It will return an error if there are still some active requests that were been submitted by the user during the session (e.g., job submission or file transfers). After the session is closed, it cannot be re-opened.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"Commands are running" [31]

7.6 changePassword

changePassword — changes the password

Synopsis

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

DESCRIPTION

This command is used to change the password. It can be done voluntarily or when the password is only temporary: for your first connection to VISHNU or after your password is reset by an administrator .

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier.

password Input argument. Password represents the password of the user.

passwordNew Input argument. PasswordNew represents the new password of the user.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Authenticator error)" [-1]

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The user is unknown or the password is wrong" [20]

"The user is locked" [23]

"You can modify information. This account is read-only" [54]

7.7 addLocalAccount

addLocalAccount — adds a new local user configuration

Synopsis

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

DESCRIPTION

A local user configuration must be added to allow you (identified by userId) to connect to machine (identified by machineId). This configuration must match an existing system account on that machine with a login that matches acLogin. The parameters sshKeyPath parameter (the absolute path to your private SSH key, used for file transfers) must be provided as well as the homeDirectory path .

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

newAccount Input argument. NewAccount is the object which encapsulates the new local user configuration.

sshPublicKey Output argument. The SSH public key generated by VISHNU for accessing a local account.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The machine is locked" [34]

"The machine does not exist or it is locked" [36]

"The local account already exists" [37]

"The system account login is already used by another vishnu user" [46]

7.8 updateLocalAccount

updateLocalAccount — updates a local user configuration

Synopsis

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

DESCRIPTION

The local user configuration can be updated. You can modify information of its local configuration such as acLogin, sshKeypath or homeDirectory.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

LocalAccUpd Input argument. Is an object which encapsulates the local user configuration changes except the machineId and the userId.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The system account login is already used by another vishnu user" [46]

7.9 deleteLocalAccount

deleteLocalAccount — removes a local user configuration (for a given user on a given machine) from VISHNU

Synopsis

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

DESCRIPTION

The local user configuration can be deleted from VISHNU. When a local user configuration is deleted, all the information about it is deleted from VISHNU.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

userId Input argument. UserId represents the VISHNU user identifier of the user whose local configuration will be deleted for the given machine .

machineId Input argument. MachineId represents the identifier of the machine whose local configuration will be deleted for the given user .

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The local account is unknown" [38]

7.10 listLocalAccounts

listLocalAccounts — lists the local user configurations

Synopsis

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

DESCRIPTION

A local configuration is used to configure the access to a given system for a given user through VISHNU. It is related to an account on that system that is identified using its login. This command allows you to check all the local configurations related to your VISHNU account.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listLocalAcct Output argument. ListLocalAccount is the list of the local user configuations .

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

7.11 listMachines

listMachines — lists the machines that are accessible through VISHNU

Synopsis

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

DESCRIPTION

This command is used to display the machines that you can use for VISHNU services. The machines you can access through VISHNU are those that are configured in VISHNU by the VISHNU administrator, and that have been added to your personal VISHNU configuration using the vishnu_add_local_account command. The results contain, for each machine, a machine identifier that you can use as a parameter for other VISHNU commands.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listMachine Output argument. ListLocalAccount is the list of the local configs .

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

7.12 listHistoryCmd

listHistoryCmd — lists the commands

Synopsis

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

DESCRIPTION

This command displays a history of the commands you ran. Several options can be used to specify which commands to list.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listCommands Output argument. ListCommands is the list of commands.

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

7.13 listOptions

listOptions — lists the options of the user

Synopsis

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

DESCRIPTION

This command displays the options you configured.

ARGUMENTS

sessionKey Input argument. The sessionKey is the identifier of the session generated by VISHNU.

listOptValues Output argument. ListOptValues is an object which encapsulates the list of options.

options Input argument. Allows the user to list a specific option or all default options values or for an admin to list options of a specific user.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The name of the user option is unknown" [41]

7.14 listSessions

listSessions — lists all sessions of the user

Synopsis

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

DESCRIPTION

This command is used to display and filter the list of all your sessions. For each session, a session identifier is provided which you can use to reconnect to a given session using the vishnu_reconnect command. The session's status is either 0 (inactive) or 1 (active).

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listsession Output argument. Listsession is the list of sessions .

options Input argument. 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.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The closure policy is unknown" [42]

7.15 configureOption

configureOption — configures an option of the user

Synopsis

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

DESCRIPTION

Options in VISHNU corresponds to the parameters of some VISHNU commands (e.g., the close policy for vishnu_connect) that can be preset in the user configuration stored by the VISHNU system. This command is used to set the value of an option for the current user (the user who opened the session).

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

optionValue Input argument. The optionValue is an object which encapsulates the option information.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The name of the user option is unknown" [41]

"The closure policy is unknown" [42]

"The value of the timeout is incorrect" [43]

"The value of the transfer command is incorrect" [44]

7.16 vishnuInitialize

vishnuInitialize — initializes VISHNU

Synopsis

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

DESCRIPTION

Calling this function is required before calling any function of the VISHNU API. It initializes the connection to the VISHNU infrastructure.

ARGUMENTS

configPath Input argument. ConfigPath is the path of VISHNU configuration file.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Internal Error: Undefined exception" [9]

7.17 vishnuFinalize

vishnuFinalize — allows a user to go out properly from VISHNU

Synopsis

int **vishnu::vishnuFinalize**();

DESCRIPTION

Calling this function is necessary to free ressources consumed due to the VISHNU API

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Internal Error: Undefined exception" [9]

7.18 listAuthSystems

listAuthSystems — lists VISHNU user-authentification systems

Synopsis

int **vishnu::listAuthSystems**(const string& sessionKey, ListAuthSystems& listAuthSys, const ListAuthSysOptions& options = ListAuthSysOptions());

DESCRIPTION

This command allows to display all user-authentication systems. By default, the user-authentication systems where the user has a local user-authentication config are listed.

ARGUMENTS

sessionKey Input argument. The sessionKey is the identifier of the session generated by VISHNU.

listAuthSys Output argument. ListAuthSys is the list of the user-authentication systems.

options Input argument. Allows an admin to list all user-authentication systems used by a specific user or a user to list all user-authentication systems declared in VISHNU (and not only those where a local user-authentication configs are defined). It also allows to list user-authentication systems of a specific type.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The type of the user-authentication system is unknown" [47]

7.19 addAuthAccount

addAuthAccount — adds a new local user-authentication configuration

Synopsis

int **vishnu::addAuthAccount**(const string& sessionKey, const AuthAccount& authAccount);

DESCRIPTION

This command allows to add a local user-authentication configuration in VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authAccount Input argument. Is an object which encapsulates the information of the local user-authentication configuration which will be added in VISHNU.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The system account login is already used by another vishnu user" [46]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account already exists" [51]

7.20 updateAuthAccount

updateAuthAccount — updates a local user-authentication configuration

Synopsis

int **vishnu::updateAuthAccount**(const string& sessionKey, const AuthAccount& authAccount);

DESCRIPTION

This command allows to update a local user-authentication configuration in VISHNU. Only the local user's login in the user-authentication system can be updated.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authAccount Input argument. Is an object which encapsulates the information of the local user-authentication configuration which will be updated.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

7.21 deleteAuthAccount

deleteAuthAccount — removes a local user-authentication configuration from VISHNU

Synopsis

int **vishnu::deleteAuthAccount**(const string& sessionKey, const string& authSystemId, const string& userId = "");

DESCRIPTION

This command allows to remove a local user-authentication configuration from VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authSystemId Input argument. AuthSystemId is the identifier of the user-authentication system.

userId Input argument. Is an admin option which represents the VISHNU identifier of the user whose local user-authentication configuration will be deleted.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

7.22 listAuthAccounts

listAuthAccounts — lists local user-authentication configurations

Synopsis

int **vishnu::listAuthAccounts**(const string& sessionKey, ListAuthAccounts& listAuthAccounts, const ListAuthAccOptions& options = ListAuthAccOptions());

DESCRIPTION

This command allows to list local user-authentication configurations of VISHNU.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listAuthAccounts Output argument. Is the list of the local user-authentication configurations.

options Input argument. Allows an admin to list all local user-authentication configurations or to list local user-authentication configurations of a specific user or for a user to list local user-authentication configuration defined for a specific user-authentication system.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The userId is unknown" [21]

"The user is locked" [23]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The sessionKey is expired. The session is closed." [29]

"The user-authentication system is unknown or locked" [48]

"The user-authentication account is unknown" [52]

Chapter 8

TMS C++ API Reference

8.1 submitJob

submitJob — Allows to submit a job consisting in running a given script on a machine.

Synopsis

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

DESCRIPTION

This command is used to submit a job to the specific batch scheduler associated to a machine. It allows describing a job in a script, using either the batch scheduler's directives or VISHNU's generic directives for all batch schedulers. If the machine identifier is equal to autom, the job will be automatically submitted on a best machine (for now three criterions are used: minimum number of waiting jobs, minimum number of running jobs and the total number of jobs) through the use of a script (scriptFilePath) which must be generic script using VISHNU's generic directives for all batch schedulers

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job must be submitted.

scriptFilePath Input argument. The path to the file containing the characteristics (job command, and batch scheduler directive required or optional) of the job to submit.

jobInfo Output argument. The Job object containing the output information (ex: jobId and jobPath) of the job to submit.

options Input argument. Is an instance of the class SubmitOptions. Each optional 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 optional values set by the options object and optional values defined in the scriptFilePath, but optional 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 optional value takes precedence over earlier occurance.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"Error invalid parameters" [10]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

8.2 getJobInfo

getJobInfo — gets information on a job from its id

Synopsis

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

DESCRIPTION

This command allows getting information about a specific job. It can return the job's status, for example.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job is running.

jobId Input argument. The id of the job .

jobInfos Output argument. The resulting information on the job.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

8.3 getJobProgress

getJobProgress — gets the progression status of jobs

Synopsis

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

DESCRIPTION

This command allows getting the progression status of a job based on the wall-clock time specified.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine to get the jobs progression.

listProgress Output argument. Is the object containing jobs progression information.

options Input argument. Is an object containing the available options jobs for progression.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

8.4 listQueues

listQueues — gets queues information

Synopsis

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

DESCRIPTION

This command displays the status of the queues of a specific machine's batch scheduler.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine that the user wants to list queues.

listofQueues Output argument. The list of queues.

queueName Input argument. If it is given, listQueues gives information only of this queue.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

8.5 listJobs

listJobs — gets a list of all submitted jobs on a machine.

Synopsis

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

DESCRIPTION

This command allows displaying the jobs submitted on a specific machine's batch scheduler. If machine identifier is equal to all, submitted jobs on all machines are listed

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the jobs are running.

listOfJobs Output argument. The constructed object list of jobs.

options Input argument. Additional options for jobs listing.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

8.6 getJobOutput

getJobOutput — gets standard output and error output files of a job given its id

Synopsis

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

DESCRIPTION

This command allows getting a job's output files.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Gets outputPath and errorPath of a job from its id.

jobId Input argument. The Id of the job.

outputInfo Output argument. The Job object containing the job output information (ex: outputPath and errorPath) of the job to submit.

outDir Input argument. The output directory where the files will be stored (default is current directory).

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

"The job is not terminated" [107]

"The job is already downloaded" [108]

8.7 getCompletedJobsOutput

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

Synopsis

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

DESCRIPTION

This command allows getting the output files of all the completed jobs.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the jobs are been submitted.

listOfResults Output argument. Is the list of jobs results.

outDir Input argument. Specifies the output directory where the files will be stored (by default, the current directory).

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

8.8 cancelJob

cancelJob — Allows to cancel a job submitted on a given machine.

Synopsis

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

DESCRIPTION

cancels a job from its id. If job id is equal to all, all submitted jobs by all users will be cancelled if the user is an administrator, and only jobs submitted by the user will be cancelled if the user is not an administrator.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job is running.

jobId Input argument. The Id of the job.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

"The job is already terminated" [105]

"The job is already canceled" [106]

8.9 addWork

addWork — Allows to create a work consisting in running jobs.

Synopsis

int **vishnu::addWork**(const string& sessionKey, Job& work);

DESCRIPTION

This command adds a work in vishnu

ARGUMENTS

sessionKey Input argument. The session key.

work Input/Output argument. The work to add.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Vishnu not available (Database connection)" [3]

"Vishnu not available (System)" [4]

"Vishnu not available (SSH error)" [9]

"Error invalid parameters" [10]

"The sessionKey is expired. The session is closed." [29]

"The machine id is unknown" [32]

"The batch scheduler type is unknown" [101]

"The batch scheduler indicates an error" [102]

"Permission denied" [104]

Chapter 9

FMS C++ API Reference

9.1 createFile

createFile — creates files on remote machines.

Synopsis

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

DESCRIPTION

Creates an empty file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to create following the pattern [host:]file path.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.2 createDir

createDir — creates directories on remote machines.

Synopsis

int **vishnu::createDir**(const string& sessionKey, const string& path, const CreateDirOptions& options = CreateDirOptions());

DESCRIPTION

Creates an new directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to create following the pattern [host:]directory path.

options Input argument. The create directory command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.3 removeFile

removeFile — removes files from remote hosts.

Synopsis

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

DESCRIPTION

Deletes a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to remove following the pattern [host:]file path.

options Input argument. The remove command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.4 removeDir

removeDir — removes directories (and subdirectories) from remote machines.

Synopsis

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

DESCRIPTION

Deletes a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to remove following the pattern [host:]directory path.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.5 chGrp

chGrp — changes group owner of remote files/directories.

Synopsis

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

DESCRIPTION

Changes the group attribute of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

group Input argument. The new group owner of file/directory.

path Input argument. The file/directory following the pattern [host:]file path.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.6 chMod

chMod — changes access rights of remote files/directories.

Synopsis

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

DESCRIPTION

Changes the permissions of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command). The mode parameter is the same value as for the unix chmod command.

ARGUMENTS

sessionKey Input argument. The session key.

mode Input argument. the access rigths of file/directory in octal system.

path Input argument. The file/directory following the pattern [host:]file path.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.7 headOfFile

headOfFile — displays a few first lines of files located on remote machines.

Synopsis

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

DESCRIPTION

Displays the first lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file following the pattern [host:]file path.

fileContent Output argument. The first "nLine" lines of the file.

options Input argument. The head commandoptions.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.8 tailOfFile

tailOfFile — displays a few last lines of files located on remote machines

Synopsis

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

DESCRIPTION

Displays the last lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file following the pattern [host:]file path.

fileContent Output argument. The last "nLine" lines of the file.

options Input argument. The tail command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.9 contentOfFile

contentOfFile — displays content of files located on remote machines

Synopsis

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

DESCRIPTION

Displays the content of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to display following the pattern [host:]file path.

fileContent Output argument. The content of the file.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.10 listDir

listDir — displays the content of a remote directory

Synopsis

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

DESCRIPTION

Displays the content of a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to list following the pattern [host:]directory path.

dirContent Output argument. The content of the directory.

options Input argument. List of options for the listDir command.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.11 copyFile

copyFile — executes a synchronous copy of file.

Synopsis

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

DESCRIPTION

Copy a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to copy following the pattern [host:]file path.

dest Input argument. The path of the destination file.

options Input argument. The copy options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.12 copyAsyncFile

copyAsyncFile — executes an asynchronous copy of file.

Synopsis

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

DESCRIPTION

Initiates a copy of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to copy following the pattern [host:]file path.

dest Input argument. The path of the destination file.

transferInfo Output argument. A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it).

options Input argument. The copy options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.13 moveFile

moveFile — executes a synchronous move of file.

Synopsis

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

DESCRIPTION

Move a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to move following the pattern [host:]file path.

dest Input argument. The path of the destination file.

options Input argument. The move command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.14 moveAsyncFile

moveAsyncFile — executes an asynchronous move of file.

Synopsis

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

DESCRIPTION

Initiates a move of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to move following the pattern [host:]file path.

dest Input argument. The path of the destination file.

transferInfo Output argument. A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it).

options Input argument. The transfer command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.15 stopFileTransfer

stopFileTransfer — stops an execution of a set of file transfers.

Synopsis

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

DESCRIPTION

Cancels a file or directory transfer that has been initiated using a vishnu asynchronous copy or move file command. The command listFileTransfers can be used to check the status of the transfer after it has been cancelled, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

options Input argument. The stop file transfer command options.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.16 listFileTransfers

listFileTransfers — displays the history of all file transfers submitted by User.

Synopsis

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

DESCRIPTION

Get the list of all file or directory transfers that have been initiated using a vishnu synchronous or asynchronous copy or move file command.

ARGUMENTS

sessionKey Input argument. The session key.

fileTransferList Output argument. The file transfer list.

options Input argument. The filter options .

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The userId is unknown" [21]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

9.17 getFileInfo

getFileInfo — displays the information of files.

Synopsis

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

DESCRIPTION

Get the details of a remote file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file whose inode information will be displayed.

filesinfo Output argument. The inode information.

EXCEPTIONS

The following exceptions may be thrown:

"Vishnu not available (Service bus failure)" [1]

"Vishnu not available (Database error)" [2]

"Internal Error: Undefined exception" [9]

"an option or a parameter provided is invalid for this service" [10]

"Undefined configuration parameter" [12]

"The user is not an administrator" [25]

"The session key is unrecognized" [28]

"The session key has expired. The session is closed." [29]

"The machine id is unknown" [32]

"The local account is unknown" [38]

"The path provided is invalid." [201]

"Runtime error" [202]

"The transfer id is unknown" [203]

Chapter 10

IMS C++ API Reference

10.1 exportCommands

exportCommands — exports all the commands made by a user during a session

Synopsis

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

DESCRIPTION

Exports all the VISHNU commands submitted during a completed session. This session must be in closed state. The output of this command is a file containing a shell script. For safety reasons, the commands having a password for parameter are not exported (for example the vishnu_connect and vishnu_change_password commands). This means the shell script must be run after opening a session manually or by adding the vishnu_connect command to the script. The access to other user's sessions is only permitted to administrators.

ARGUMENTS

sessionKey Input argument. The session key.

oldSessionId Input argument. The id of the session to export (session has ended).

filename Input/Output argument. The path of the output file containing the Vishnu shell commands.

options Input argument. Options which encapsulate the option for the export.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

10.2 getMetricCurrentValue

getMetricCurrentValue — displays the current values of system metrics

Synopsis

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

DESCRIPTION

Displays the current values of the monitored metrics on the system identified by the machineId argument : cpuload, free diskspace and free memory. The units of displayed values are percentages for cpuload and Megabytes (Mb) for diskspace and memory. The provided values are always standard integers (no float values). Please note that retrieving these values uses some valuable system ressources and should not occur too frequently to avoid an impact on system performance.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine.

metricValue Output argument. Value of the metric.

options Input argument. The options for the current metric value.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

10.3 getMetricHistory

getMetricHistory — displays the history of values of a system metric

Synopsis

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

DESCRIPTION

Displays the chronological list of values of the metrics on the system identified by the machineId argument. Using the options it is possible to specify a type of metric and the starting and ending dates of the desired monitoring period. Note that some data will be available only if the required VISHNU agent (IMS server) has been running locally on the machine during the specified period.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine.

metricValues Output argument. List of metric values.

options Input argument. The optional fields for the metric history.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

10.4 getUpdateFrequency

getUpdateFrequency — gets the update frequency of the IMS database

Synopsis

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

DESCRIPTION

This function allows a user to get the update frequency, to know how often the state of the machines is automatically polled to get historical data.

ARGUMENTS

sessionKey Input argument. The session key.

freq Output argument. Frequency the data are updated, in second.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

10.5 getSystemInfo

getSystemInfo — To get the system info on a machine

Synopsis

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

DESCRIPTION

This function allows a user to get system information about a machine. A system information describes a machine. The option is the machine id (if no machine id, the information for all the machines are given)

ARGUMENTS

sessionKey Input argument. The session key.

res Output argument. The list of the system information gotten.

options Input argument. Optional field for system information.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"Undefined error code" [9]

"If a parameter is invalid" [10]

"The session key is unrecognized." [28]

"The session key has expired. The session is closed." [29]

Chapter 11

UMS Python API Reference

11.1 VISHNU.connect

VISHNU.connect — opens a session

Synopsis

ret=VISHNU.connect(string userId, string password, Session session, ConnectOptions options = ConnectOptions());

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. The userId and password may be given through the options, or if no userId and password are specified, vishnu_connect will read them in the .netrc file located in the home of the user.

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier. If userId and password are empty, vishnu connect will read them in the .netrc file located in the home of the user.

password Input argument. Password represents the password of the user. If userId and password are empty, vishnu connect will read them in the .netrc file located in the home of the user.

session Output argument. The session object that contains the created session details.

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Authenticator error)" [-1])

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The user is unknown or the password is wrong" [20])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The closure policy is unknown" [42])

UMSVishnuException("The value of the timeout is incorrect" [43])

11.2 VISHNU.connect

VISHNU.connect — opens a session by trying multiple couples (userId, password) each in turn

Synopsis

ret=VISHNU.connect(ListUsers listUsers, Session session, ConnectOptions options = ConnectOptions());

DESCRIPTION

Opening a VISHNU session is the first step before using any other VISHNU command. This command authenticates you. You must have been registered in the VISHNU system by an administrator. It also creates a session that remains open after the command is completed and until the session is either manually or automatically closed. It is possible to define multiple VISHNU couples (userId, password). In this case, vishnu_connect_m read them and tries each couple to make a connection.

ARGUMENTS

listUsers Input argument. The list containing one or more couple (userId and password) potentially usable for connection.

session Output argument. The session object that contains the created session details.

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Authenticator error)" [-1])

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The user is unknown or the password is wrong" [20])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The closure policy is unknown" [42])

UMSVishnuException("The value of the timeout is incorrect" [43])

11.3 VISHNU.reconnect

VISHNU.reconnect — reconnects to a session that is still active

Synopsis

ret=VISHNU.reconnect(string userId, string password, string sessionId, Session session);

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The userId and password may be given through the options, or if no userId and password are specified, vishnu_reconnect will read them in the .netrc file located in the home of the user.

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier. If userId and password are empty, vishnu reconnect will read them in the .netrc file located in the home of the user.

password Input argument. Password represents the password of the user. If userId and password are empty, vishnu reconnect will read them in the .netrc file located in the home of the user.

sessionId Input argument. SessionId is the identifier of the session defined in the database.

session Output argument. The session object containing session information.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Authenticator error)" [-1])

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The user is unknown or the password is wrong" [20])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The session Id is unknown" [30])

UMSVishnuException("The machine does not exist or it is locked" [36])

11.4 VISHNU.reconnect

VISHNU.reconnect — reconnects to a session that is still active by trying multiple couples (userId, password) each in turn

Synopsis

ret=VISHNU.reconnect(ListUsers listUsers, string sessionId, Session session);

DESCRIPTION

This command allows you to resume a session that has been opened previously and that has not yet been closed. You can disconnect from a session without closing it (for example if there are running commands in that session) by setting the session's close policy (at connection time) to CLOSE_ON_TIMEOUT. As sessions are linked to a specific client system, you cannot reconnect to a session that was opened on another client system. The vishnu_reconnect_m tries in turn multiple couples (userId, password).

ARGUMENTS

listUsers Input argument. The list containing one or more couple (userId and password) potentially usable for connection.

sessionId Input argument. SessionId is the identifier of the session defined in the database.

session Output argument. The session object containing session information.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Authenticator error)" [-1])

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The user is unknown or the password is wrong" [20])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The session Id is unknown" [30])

UMSVishnuException("The machine does not exist or it is locked" [36])

11.5 VISHNU.close

VISHNU.close — closes the session

Synopsis

ret=VISHNU.close(string sessionKey);

DESCRIPTION

This command closes the session that is currently active in the terminal. It will return an error if there are still some active requests that were been submitted by the user during the session (e.g., job submission or file transfers). After the session is closed, it cannot be re-opened.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("Commands are running" [31])

11.6 VISHNU.changePassword

VISHNU.changePassword — changes the password

Synopsis

ret=VISHNU.changePassword(string userId, string password, string passwordNew);

DESCRIPTION

This command is used to change the password. It can be done voluntarily or when the password is only temporary: for your first connection to VISHNU or after your password is reset by an administrator .

ARGUMENTS

userId Input argument. UserId represents the VISHNU user identifier.

password Input argument. Password represents the password of the user.

passwordNew Input argument. PasswordNew represents the new password of the user.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Authenticator error)" [-1])

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The user is unknown or the password is wrong" [20])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("You can modify information. This account is read-only" [54])

11.7 VISHNU.addLocalAccount

VISHNU.addLocalAccount — adds a new local user configuration

Synopsis

ret, sshPublicKey=VISHNU.addLocalAccount(string sessionKey, LocalAccount newAccount);

DESCRIPTION

A local user configuration must be added to allow you (identified by userId) to connect to machine (identified by machineId). This configuration must match an existing system account on that machine with a login that matches acLogin. The parameters sshKeyPath parameter (the absolute path to your private SSH key, used for file transfers) must be provided as well as the homeDirectory path .

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

newAccount Input argument. NewAccount is the object which encapsulates the new local user configuration.

sshPublicKey Output argument. The SSH public key generated by VISHNU for accessing a local account.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

sshPublicKey(string) The SSH public key generated by VISHNU for accessing a local account

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The machine is locked" [34])

UMSVishnuException("The machine does not exist or it is locked" [36])

UMSVishnuException("The local account already exists" [37])

UMSVishnuException("The system account login is already used by another vishnu user" [46])

11.8 VISHNU.updateLocalAccount

VISHNU.updateLocalAccount — updates a local user configuration

Synopsis

ret=VISHNU.updateLocalAccount(string sessionKey, LocalAccount LocalAccUpd);

DESCRIPTION

The local user configuration can be updated. You can modify information of its local configuration such as acLogin, sshKeypath or homeDirectory.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

LocalAccUpd Input argument. Is an object which encapsulates the local user configuration changes except the machineId and the userId.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

UMSVishnuException("The system account login is already used by another vishnu user" [46])

11.9 VISHNU.deleteLocalAccount

VISHNU.deleteLocalAccount — removes a local user configuration (for a given user on a given machine) from VISHNU

Synopsis

ret=VISHNU.deleteLocalAccount(string sessionKey, string userId, string machineId);

DESCRIPTION

The local user configuration can be deleted from VISHNU. When a local user configuration is deleted, all the information about it is deleted from VISHNU.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

userId Input argument. UserId represents the VISHNU user identifier of the user whose local configuration will be deleted for the given machine .

machineId Input argument. MachineId represents the identifier of the machine whose local configuration will be deleted for the given user .

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The local account is unknown" [38])

11.10 VISHNU.listLocalAccounts

VISHNU.listLocalAccounts — lists the local user configurations

Synopsis

ret, listLocalAcct=VISHNU.listLocalAccounts(string sessionKey, ListLocalAccOptions options = ListLocalAccOptions());

DESCRIPTION

A local configuration is used to configure the access to a given system for a given user through VISHNU. It is related to an account on that system that is identified using its login. This command allows you to check all the local configurations related to your VISHNU account.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listLocalAcct Output argument. ListLocalAccount is the list of the local user configuations .

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

11.11 VISHNU.listMachines

VISHNU.listMachines — lists the machines that are accessible through VISHNU

Synopsis

ret, listMachine=VISHNU.listMachines(string sessionKey, ListMachineOptions options = ListMachineOptions());

DESCRIPTION

This command is used to display the machines that you can use for VISHNU services. The machines you can access through VISHNU are those that are configured in VISHNU by the VISHNU administrator, and that have been added to your personal VISHNU configuration using the vishnu_add_local_account command. The results contain, for each machine, a machine identifier that you can use as a parameter for other VISHNU commands.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listMachine Output argument. ListLocalAccount is the list of the local configs .

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

11.12 VISHNU.listHistoryCmd

VISHNU.listHistoryCmd — lists the commands

Synopsis

ret, listCommands=VISHNU.listHistoryCmd(string sessionKey, ListCmdOptions options = ListCmdOptions());

DESCRIPTION

This command displays a history of the commands you ran. Several options can be used to specify which commands to list.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listCommands Output argument. ListCommands is the list of commands.

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

11.13 VISHNU.listOptions

VISHNU.listOptions — lists the options of the user

Synopsis

ret, listOptValues=VISHNU.listOptions(string sessionKey, ListOptOptions options = ListOptOptions());

DESCRIPTION

This command displays the options you configured.

ARGUMENTS

sessionKey Input argument. The sessionKey is the identifier of the session generated by VISHNU.

listOptValues Output argument. ListOptValues is an object which encapsulates the list of options.

options Input argument. Allows the user to list a specific option or all default options values or for an admin to list options of a specific user.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The name of the user option is unknown" [41])

11.14 VISHNU.listSessions

VISHNU.listSessions — lists all sessions of the user

Synopsis

ret, listsession=VISHNU.listSessions(string sessionKey, ListSessionOptions options = ListSessionOptions());

DESCRIPTION

This command is used to display and filter the list of all your sessions. For each session, a session identifier is provided which you can use to reconnect to a given session using the vishnu_reconnect command. The session's status is either 0 (inactive) or 1 (active).

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listsession Output argument. Listsession is the list of sessions .

options Input argument. 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.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The closure policy is unknown" [42])

11.15 VISHNU.configureOption

VISHNU.configureOption — configures an option of the user

Synopsis

ret=VISHNU.configureOption(string sessionKey, OptionValue optionValue);

DESCRIPTION

Options in VISHNU corresponds to the parameters of some VISHNU commands (e.g., the close policy for vishnu_connect) that can be preset in the user configuration stored by the VISHNU system. This command is used to set the value of an option for the current user (the user who opened the session).

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

optionValue Input argument. The optionValue is an object which encapsulates the option information.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The name of the user option is unknown" [41])

UMSVishnuException("The closure policy is unknown" [42])

UMSVishnuException("The value of the timeout is incorrect" [43])

UMSVishnuException("The value of the transfer command is incorrect" [44])

11.16 VISHNU.vishnuInitialize

VISHNU.vishnuInitialize — initializes VISHNU

Synopsis

ret=VISHNU.vishnuInitialize(string configPath);

DESCRIPTION

Calling this function is required before calling any function of the VISHNU API. It initializes the connection to the VISHNU infrastructure.

ARGUMENTS

configPath Input argument. ConfigPath is the path of VISHNU configuration file.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Internal Error: Undefined exception" [9])

11.17 VISHNU.vishnuFinalize

VISHNU.vishnuFinalize — allows a user to go out properly from VISHNU

Synopsis

ret=VISHNU.vishnuFinalize();

DESCRIPTION

Calling this function is necessary to free ressources consumed due to the VISHNU API

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Internal Error: Undefined exception" [9])

11.18 VISHNU.listAuthSystems

VISHNU.listAuthSystems — lists VISHNU user-authentification systems

Synopsis

ret, listAuthSys=VISHNU.listAuthSystems(string sessionKey, ListAuthSysOptions options = ListAuthSysOptions());

DESCRIPTION

This command allows to display all user-authentication systems. By default, the user-authentication systems where the user has a local user-authentication config are listed.

ARGUMENTS

sessionKey Input argument. The sessionKey is the identifier of the session generated by VISHNU.

listAuthSys Output argument. ListAuthSys is the list of the user-authentication systems.

options Input argument. Allows an admin to list all user-authentication systems used by a specific user or a user to list all user-authentication systems declared in VISHNU (and not only those where a local user-authentication configs are defined). It also allows to list user-authentication systems of a specific type.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The type of the user-authentication system is unknown" [47])

11.19 VISHNU.addAuthAccount

VISHNU.addAuthAccount — adds a new local user-authentication configuration

Synopsis

ret=VISHNU.addAuthAccount(string sessionKey, AuthAccount authAccount);

DESCRIPTION

This command allows to add a local user-authentication configuration in VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authAccount Input argument. Is an object which encapsulates the information of the local user-authentication configuration which will be added in VISHNU.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The system account login is already used by another vishnu user" [46])

UMSVishnuException("The user-authentication system is unknown or locked" [48])

UMSVishnuException("The user-authentication account already exists" [51])

11.20 VISHNU.updateAuthAccount

VISHNU.updateAuthAccount — updates a local user-authentication configuration

Synopsis

ret=VISHNU.updateAuthAccount(string sessionKey, AuthAccount authAccount);

DESCRIPTION

This command allows to update a local user-authentication configuration in VISHNU. Only the local user's login in the user-authentication system can be updated.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authAccount Input argument. Is an object which encapsulates the information of the local user-authentication configuration which will be updated.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The user-authentication system is unknown or locked" [48])

UMSVishnuException("The user-authentication account is unknown" [52])

11.21 VISHNU.deleteAuthAccount

VISHNU.deleteAuthAccount — removes a local user-authentication configuration from VISHNU

Synopsis

ret=VISHNU.deleteAuthAccount(string sessionKey, string authSystemId, string userId = "");

DESCRIPTION

This command allows to remove a local user-authentication configuration from VISHNU. The required parameters are : the VISHNU user identifier, the user-authentication system identifier and the uid of the user on the corresponding user-authentication system

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

authSystemId Input argument. AuthSystemId is the identifier of the user-authentication system.

userId Input argument. Is an admin option which represents the VISHNU identifier of the user whose local user-authentication configuration will be deleted.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The user-authentication system is unknown or locked" [48])

UMSVishnuException("The user-authentication account is unknown" [52])

11.22 VISHNU.listAuthAccounts

VISHNU.listAuthAccounts — lists local user-authentication configurations

Synopsis

ret, listAuthAccounts=VISHNU.listAuthAccounts(string sessionKey, ListAuthAccOptions options = ListAuthAccOptions());

DESCRIPTION

This command allows to list local user-authentication configurations of VISHNU.

ARGUMENTS

sessionKey Input argument. The sessionKey is the encrypted identifier of the session generated by VISHNU.

listAuthAccounts Output argument. Is the list of the local user-authentication configurations.

options Input argument. Allows an admin to list all local user-authentication configurations or to list local user-authentication configurations of a specific user or for a user to list local user-authentication configuration defined for a specific user-authentication system.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException(''Vishnu not available (Service bus failure)'' [1])

SystemException(''Vishnu not available (Database error)'' [2])

SystemException(''Vishnu not available (Database connection)'' [3])

SystemException(''Vishnu not available (System)'' [4])

SystemException(''Internal Error: Undefined exception'' [9])

UMSVishnuException(''The userId is unknown'' [21])

UMSVishnuException(''The user is locked'' [23])

UMSVishnuException(''The user is not an administrator'' [25])

UMSVishnuException(''The session key is unrecognized'' [28])

UMSVishnuException(''The sessionKey is expired. The session is closed.'' [29])

UMSVishnuException(''The user-authentication system is unknown or locked'' [48])

UMSVishnuException(''The user-authentication account is unknown'' [52])

Chapter 12

TMS Python API Reference

12.1 VISHNU.submitJob

VISHNU.submitJob — Allows to submit a job consisting in running a given script on a machine.

Synopsis

ret=VISHNU.submitJob(string sessionKey, string machineId, string scriptFilePath, Job jobInfo, SubmitOptions options = SubmitOptions());

DESCRIPTION

This command is used to submit a job to the specific batch scheduler associated to a machine. It allows describing a job in a script, using either the batch scheduler's directives or VISHNU's generic directives for all batch schedulers. If the machine identifier is equal to autom, the job will be automatically submitted on a best machine (for now three criterions are used: minimum number of waiting jobs, minimum number of running jobs and the total number of jobs) through the use of a script (scriptFilePath) which must be generic script using VISHNU's generic directives for all batch schedulers

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job must be submitted.

scriptFilePath Input argument. The path to the file containing the characteristics (job command, and batch scheduler directive required or optional) of the job to submit.

jobInfo Output argument. The Job object containing the output information (ex: jobId and jobPath) of the job to submit.

options Input argument. Is an instance of the class SubmitOptions. Each optional value is associated to a set operation (e.g: setNbCpu(...)) in the class SubmitOptions. If no set operation is not called on the instance object options, the job is submitted with the options defined in the scriptFilePath. Otherwise the job is submitted with the optional values set by the options object and optional values defined in the scriptFilePath, but optional 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 optional value takes precedence over earlier occurance.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Vishnu not available (SSH error)" [9])

UserException("Error invalid parameters" [10])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

12.2 VISHNU.getJobInfo

VISHNU.getJobInfo — gets information on a job from its id

Synopsis

ret=VISHNU.getJobInfo(string sessionKey, string machineId, string jobId, Job jobInfos);

DESCRIPTION

This command allows getting information about a specific job. It can return the job's status, for example.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job is running.

jobId Input argument. The id of the job .

jobInfos Output argument. The resulting information on the job.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

12.3 VISHNU.getJobProgress

VISHNU.getJobProgress — gets the progression status of jobs

Synopsis

ret, listProgress=VISHNU.getJobProgress(string sessionKey, string machineId, ProgressOptions options = ProgressOptions());

DESCRIPTION

This command allows getting the progression status of a job based on the wall-clock time specified.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine to get the jobs progression.

listProgress Output argument. Is the object containing jobs progression information.

options Input argument. Is an object containing the available options jobs for progression.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

12.4 VISHNU.listQueues

VISHNU.listQueues — gets queues information

Synopsis

ret, listofQueues=VISHNU.listQueues(string sessionKey, string machineId, string queueName = string());

DESCRIPTION

This command displays the status of the queues of a specific machine's batch scheduler.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine that the user wants to list queues.

listofQueues Output argument. The list of queues.

queueName Input argument. If it is given, listQueues gives information only of this queue.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

12.5 VISHNU.listJobs

VISHNU.listJobs — gets a list of all submitted jobs on a machine.

Synopsis

ret, listOfJobs=VISHNU.listJobs(string sessionKey, string machineId, ListJobsOptions options = ListJobsOptions());

DESCRIPTION

This command allows displaying the jobs submitted on a specific machine's batch scheduler. If machine identifier is equal to all, submitted jobs on all machines are listed

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the jobs are running.

listOfJobs Output argument. The constructed object list of jobs.

options Input argument. Additional options for jobs listing.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

12.6 VISHNU.getJobOutput

VISHNU.getJobOutput — gets standard output and error output files of a job given its id

Synopsis

ret=VISHNU.getJobOutput(string sessionKey, string machineId, string jobId, JobResult outputInfo, string outDir = string());

DESCRIPTION

This command allows getting a job's output files.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Gets outputPath and errorPath of a job from its id.

jobId Input argument. The Id of the job.

outputInfo Output argument. The Job object containing the job output information (ex: outputPath and errorPath) of the job to submit.

outDir Input argument. The output directory where the files will be stored (default is current directory).

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Vishnu not available (SSH error)" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

TMSVishnuException("The job is not terminated" [107])

TMSVishnuException("The job is already downloaded" [108])

12.7 VISHNU.getCompletedJobsOutput

VISHNU.getCompletedJobsOutput — gets standard output and error output files of completed jobs (applies only once for each job)

Synopsis

ret, listOfResults=VISHNU.getCompletedJobsOutput(string sessionKey, string machineId, string outDir = string());

DESCRIPTION

This command allows getting the output files of all the completed jobs.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the jobs are been submitted.

listOfResults Output argument. Is the list of jobs results.

outDir Input argument. Specifies the output directory where the files will be stored (by default, the current directory).

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Vishnu not available (SSH error)" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

12.8 VISHNU.cancelJob

VISHNU.cancelJob — Allows to cancel a job submitted on a given machine.

Synopsis

ret=VISHNU.cancelJob(string sessionKey, string machineId, string jobId);

DESCRIPTION

cancels a job from its id. If job id is equal to all, all submitted jobs by all users will be cancelled if the user is an administrator, and only jobs submitted by the user will be cancelled if the user is not an administrator.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. Is the id of the machine on which the job is running.

jobId Input argument. The Id of the job.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Vishnu not available (Database connection)" [3])

SystemException("Vishnu not available (System)" [4])

SystemException("Vishnu not available (SSH error)" [9])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

TMSVishnuException("The batch scheduler type is unknown" [101])

TMSVishnuException("The batch scheduler indicates an error" [102])

TMSVishnuException("Permission denied" [104])

TMSVishnuException("The job is already terminated" [105])

TMSVishnuException("The job is already canceled" [106])

12.9 VISHNU.addWork

VISHNU.addWork — Allows to create a work consisting in running jobs.

Synopsis

ret=VISHNU.addWork(string sessionKey, Job work);

DESCRIPTION

This command adds a work in vishnu

ARGUMENTS

sessionKey Input argument. The session key.

work Input/Output argument. The work to add.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException(''Vishnu not available (Service bus failure)'' [1])

SystemException(''Vishnu not available (Database error)'' [2])

SystemException(''Vishnu not available (Database connection)'' [3])

SystemException(''Vishnu not available (System)'' [4])

SystemException(''Vishnu not available (SSH error)'' [9])

UserException(''Error invalid parameters'' [10])

UMSVishnuException(''The sessionKey is expired. The session is closed.'' [29])

UMSVishnuException(''The machine id is unknown'' [32])

TMSVishnuException(''The batch scheduler type is unknown'' [101])

TMSVishnuException(''The batch scheduler indicates an error'' [102])

TMSVishnuException(''Permission denied'' [104])

Chapter 13

FMS Python API Reference

13.1 VISHNU.createFile

VISHNU.createFile — creates files on remote machines.

Synopsis

ret=VISHNU.createFile(string sessionKey, string path);

DESCRIPTION

Creates an empty file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to create following the pattern [host:]file path.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.2 VISHNU.createDir

VISHNU.createDir — creates directories on remote machines.

Synopsis

ret=VISHNU.createDir(string sessionKey, string path, CreateDirOptions options = CreateDirOptions());

DESCRIPTION

Creates an new directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to create following the pattern [host:]directory path.

options Input argument. The create directory command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.3 VISHNU.removeFile

VISHNU.removeFile — removes files from remote hosts.

Synopsis

ret=VISHNU.removeFile(string sessionKey, string path, RmFileOptions options = RmFileOptions());

DESCRIPTION

Deletes a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to remove following the pattern [host:]file path.

options Input argument. The remove command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.4 VISHNU.removeDir

VISHNU.removeDir — removes directories (and subdirectories) from remote machines.

Synopsis

ret=VISHNU.removeDir(string sessionKey, string path);

DESCRIPTION

Deletes a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to remove following the pattern [host:]directory path.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.5 VISHNU.chGrp

VISHNU.chGrp — changes group owner of remote files/directories.

Synopsis

ret=VISHNU.chGrp(string sessionKey, string group, string path);

DESCRIPTION

Changes the group attribute of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command)

ARGUMENTS

sessionKey Input argument. The session key.

group Input argument. The new group owner of file/directory.

path Input argument. The file/directory following the pattern [host:]file path.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.6 VISHNU.chMod

VISHNU.chMod — changes access rights of remote files/directories.

Synopsis

ret=VISHNU.chMod(string sessionKey, mode_t mode, string path);

DESCRIPTION

Changes the permissions of a file or directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command). The mode parameter is the same value as for the unix chmod command.

ARGUMENTS

sessionKey Input argument. The session key.

mode Input argument. the access rigths of file/directory in octal system.

path Input argument. The file/directory following the pattern [host:]file path.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.7 VISHNU.headOfFile

VISHNU.headOfFile — displays a few first lines of files located on remote machines.

Synopsis

ret, fileContent=VISHNU.headOfFile(string sessionKey, string path, HeadOfFileOptions options = HeadOfFileOptions());

DESCRIPTION

Displays the first lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file following the pattern [host:]file path.

fileContent Output argument. The first "nLine" lines of the file.

options Input argument. The head commandoptions.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

fileContent(string) The first "nLine" lines of the file

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.8 VISHNU.tailOfFile

VISHNU.tailOfFile — displays a few last lines of files located on remote machines

Synopsis

ret, fileContent=VISHNU.tailOfFile(string sessionKey, string path, TailOfFileOptions options = TailOfFileOptions());

DESCRIPTION

Displays the last lines of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file following the pattern [host:]file path.

fileContent Output argument. The last "nLine" lines of the file.

options Input argument. The tail command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

fileContent(string) The last "nLine" lines of the file

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.9 VISHNU.contentOfFile

VISHNU.contentOfFile — displays content of files located on remote machines

Synopsis

ret, fileContent=VISHNU.contentOfFile(string sessionKey, string path);

DESCRIPTION

Displays the content of a file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file to display following the pattern [host:]file path.

fileContent Output argument. The content of the file.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

fileContent(string) The content of the file

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.10 VISHNU.listDir

VISHNU.listDir — displays the content of a remote directory

Synopsis

ret, dirContent=VISHNU.listDir(string sessionKey, string path, LsDirOptions options);

DESCRIPTION

Displays the content of a directory at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The directory to list following the pattern [host:]directory path.

dirContent Output argument. The content of the directory.

options Input argument. List of options for the listDir command.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.11 VISHNU.copyFile

VISHNU.copyFile — executes a synchronous copy of file.

Synopsis

ret=VISHNU.copyFile(string sessionKey, string src, string dest, CpFileOptions options = CpFileOptions());

DESCRIPTION

Copy a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to copy following the pattern [host:]file path.

dest Input argument. The path of the destination file.

options Input argument. The copy options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.12 VISHNU.copyAsyncFile

VISHNU.copyAsyncFile — executes an asynchronous copy of file.

Synopsis

ret=VISHNU.copyAsyncFile(string sessionKey, string src, string dest, FileTransfer transferInfo, CpFileOptions options);

DESCRIPTION

Initiates a copy of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local copy cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to copy following the pattern [host:]file path.

dest Input argument. The path of the destination file.

transferInfo Output argument. A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it).

options Input argument. The copy options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.13 VISHNU.moveFile

VISHNU.moveFile — executes a synchronous move of file.

Synopsis

ret=VISHNU.moveFile(string sessionKey, string src, string dest, CpFileOptions options);

DESCRIPTION

Move a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to move following the pattern [host:]file path.

dest Input argument. The path of the destination file.

options Input argument. The move command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.14 VISHNU.moveAsyncFile

VISHNU.moveAsyncFile — executes an asynchronous move of file.

Synopsis

ret=VISHNU.moveAsyncFile(string sessionKey, string src, string dest, FileTransfer transferInfo, CpFileOptions options = CpFileOptions());

DESCRIPTION

Initiates a move of a file or directory from the location given by the src parameter to the location given by the dest parameter. The src parameter must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command) or 'path' only if the file is on the local system. The dest parameter must be provided in a similar way. Note that one of the two locations at least must be a remote location, i.e. a local move cannot be handled by this command. The command listFileTransfers can be used to check the status of the transfer after it is initiated, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

src Input argument. The source file to move following the pattern [host:]file path.

dest Input argument. The path of the destination file.

transferInfo Output argument. A file tranfer identifier (allowing for instance to ckeck the status of a file transfer, or to cancel it).

options Input argument. The transfer command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.15 VISHNU.stopFileTransfer

VISHNU.stopFileTransfer — stops an execution of a set of file transfers.

Synopsis

ret=VISHNU.stopFileTransfer(string sessionKey, StopTransferOptions options = StopTransferOptions());

DESCRIPTION

Cancels a file or directory transfer that has been initiated using a vishnu asynchronous copy or move file command. The command listFileTransfers can be used to check the status of the transfer after it has been cancelled, using the transfer id as the identifier.

ARGUMENTS

sessionKey Input argument. The session key.

options Input argument. The stop file transfer command options.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.16 VISHNU.listFileTransfers

VISHNU.listFileTransfers — displays the history of all file transfers submitted by User.

Synopsis

ret, fileTransferList=VISHNU.listFileTransfers(string sessionKey, LsTransferOptions options = LsTransferOptions());

DESCRIPTION

Get the list of all file or directory transfers that have been initiated using a vishnu synchronous or asynchronous copy or move file command.

ARGUMENTS

sessionKey Input argument. The session key.

fileTransferList Output argument. The file transfer list.

options Input argument. The filter options .

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The userId is unknown" [21])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

13.17 VISHNU.getFileInfo

VISHNU.getFileInfo — displays the information of files.

Synopsis

ret=VISHNU.getFileInfo(string sessionKey, string path, FileStat filesinfo);

DESCRIPTION

Get the details of a remote file at the location given by the path parameter. The path must be provided in the form 'machine_id:path' (the machine_id values can be obtained using the vishnu_list_machine command).

ARGUMENTS

sessionKey Input argument. The session key.

path Input argument. The file whose inode information will be displayed.

filesinfo Output argument. The inode information.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("Vishnu not available (Service bus failure)" [1])

SystemException("Vishnu not available (Database error)" [2])

SystemException("Internal Error: Undefined exception" [9])

UserException("an option or a parameter provided is invalid for this service" [10])

UserException("Undefined configuration parameter" [12])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

UMSVishnuException("The machine id is unknown" [32])

UMSVishnuException("The local account is unknown" [38])

FMSVishnuException("The path provided is invalid." [201])

FMSVishnuException("Runtime error" [202])

FMSVishnuException("The transfer id is unknown" [203])

Chapter 14

IMS Python API Reference

14.1 VISHNU.exportCommands

VISHNU.exportCommands — exports all the commands made by a user during a session

Synopsis

ret=VISHNU.exportCommands(string sessionKey, string oldSessionId, string filename, ExportOp options = ExportOp());

DESCRIPTION

Exports all the VISHNU commands submitted during a completed session. This session must be in closed state. The output of this command is a file containing a shell script. For safety reasons, the commands having a password for parameter are not exported (for example the vishnu_connect and vishnu_change_password commands). This means the shell script must be run after opening a session manually or by adding the vishnu_connect command to the script. The access to other user's sessions is only permitted to administrators.

ARGUMENTS

sessionKey Input argument. The session key.

oldSessionId Input argument. The id of the session to export (session has ended).

filename Input/Output argument. The path of the output file containing the Vishnu shell commands.

options Input argument. Options which encapsulate the option for the export.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("The database generated an error" [2])

SystemException("Undefined error code" [9])

UserException("If a parameter is invalid" [10])

UMSVishnuException("The session key is unrecognized." [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

14.2 VISHNU.getMetricCurrentValue

VISHNU.getMetricCurrentValue — displays the current values of system metrics

Synopsis

ret, metricValue=VISHNU.getMetricCurrentValue(string sessionKey, string machineId, CurMetricOp options);

DESCRIPTION

Displays the current values of the monitored metrics on the system identified by the machineId argument : cpuload, free diskspace and free memory. The units of displayed values are percentages for cpuload and Megabytes (Mb) for diskspace and memory. The provided values are always standard integers (no float values). Please note that retrieving these values uses some valuable system ressources and should not occur too frequently to avoid an impact on system performance.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine.

metricValue Output argument. Value of the metric.

options Input argument. The options for the current metric value.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("The database generated an error" [2])

SystemException("Undefined error code" [9])

UserException("If a parameter is invalid" [10])

UMSVishnuException("The session key is unrecognized." [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

14.3 VISHNU.getMetricHistory

VISHNU.getMetricHistory — displays the history of values of a system metric

Synopsis

ret, metricValues=VISHNU.getMetricHistory(string sessionKey, string machineId, MetricHistOp options = MetricHistOp());

DESCRIPTION

Displays the chronological list of values of the metrics on the system identified by the machineId argument. Using the options it is possible to specify a type of metric and the starting and ending dates of the desired monitoring period. Note that some data will be available only if the required VISHNU agent (IMS server) has been running locally on the machine during the specified period.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine.

metricValues Output argument. List of metric values.

options Input argument. The optional fields for the metric history.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException("The database generated an error" [2])

SystemException("Undefined error code" [9])

UserException("If a parameter is invalid" [10])

UMSVishnuException("The session key is unrecognized." [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

14.4 VISHNU.getUpdateFrequency

VISHNU.getUpdateFrequency — gets the update frequency of the IMS database

Synopsis

ret, freq=VISHNU.getUpdateFrequency(string sessionKey);

DESCRIPTION

This function allows a user to get the update frequency, to know how often the state of the machines is automatically polled to get historical data.

ARGUMENTS

sessionKey Input argument. The session key.

freq Output argument. Frequency the data are updated, in second.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

freq(int) Frequency the data are updated, in second

EXCEPTIONS

The following exceptions may be thrown:

SystemException("The database generated an error" [2])

SystemException("Undefined error code" [9])

UserException("If a parameter is invalid" [10])

UMSVishnuException("The session key is unrecognized." [28])

UMSVishnuException("The session key has expired. The session is closed." [29])

14.5 VISHNU.getSystemInfo

VISHNU.getSystemInfo — To get the system info on a machine

Synopsis

ret, res=VISHNU.getSystemInfo(string sessionKey, SysInfoOp options = SysInfoOp());

DESCRIPTION

This function allows a user to get system information about a machine. A system information describes a machine. The option is the machine id (if no machine id, the information for all the machines are given)

ARGUMENTS

sessionKey Input argument. The session key.

res Output argument. The list of the system information gotten.

options Input argument. Optional field for system information.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

SystemException(''The database generated an error'' [2])

SystemException(''Undefined error code'' [9])

UserException(''If a parameter is invalid'' [10])

UMSVishnuException(''The session key is unrecognized.'' [28])

UMSVishnuException(''The session key has expired. The session is closed.'' [29])