VISHNU - The guide of the administrator

COLLABORATORS			
	TITLE : VISHNU - The guide of the administrator		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé	October 5, 2012	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1	08/03/2011	Initial version of the document, only for the UMS module	K. COULOMB
2	18/03/2011	Add a paragraph to launch manually using the forwarders and config files	K. COULOMB
3	22/03/2011	Add the web services	K. COULOMB
4	11/05/2011	Rewrite how to deploy with the config files. Add a paragraph for the sendmail function. Add the TMS data.	K. COULOMB, B.ISNARD
5	18/05/2011	Add the configuration parameter dbConnectionsNb.	B.ISNARD
6	10/06/2011	Documentation for IMS.	K.COULOMB
7	15/06/2011	Documentation for FMS.	I.CISSE
8	22/06/2011	Add the option ENABLE_SWIG.	B.ISNARD
9	24/06/2011	Add the option vishnuMachineId in the UMS, IMS and FMS configuration files.	I.CISSE

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
10	13/07/2011	Update document after the first feedbacks .	K.COULOMB
11	11/08/2011	Take SLURM into account	D.TRAORE
12	23/08/2011	Add link on ow to install postgre/mysql databases. Delete the update database part (the creation script contains everything). Add a reference to 'VISHNU_API'	K.COULOMB
13	14/12/2011	Update for the new dietForwarder	K.COULOMB
14	15/12/2011	Update the section concerning the configuration of the SSH keys for FMS.	I.CISSE
15	16/12/2011	Add the section to configure the SSH keys in TMS.	D.TRAORE
16	30/01/2012	Update the DIET version dependency.	K. COULOMB
17	27/02/2012	Add the list of the boost libraries necessaries.	K. COULOMB
18	02/03/2012	Add the LSF batch	D.TRAORE
19	22/03/2012	Add the LDAP support	K. COULOMB
20	11/04/2012	Add the Grid Engine batch handling	E. PAMBA CAPO-CHICHI
21	30/05/2012	Add version of LoadLeveler, GLIBC for libcrypt and ssh	E. PAMBA CAPO-CHICHI
22	27/08/2012	Add new TMS compilation documentation	K. COULOMB
23	01/10/2012	Adding a prerequisites section and refining of the installation process	K. Chakode

Contents

Chapter 1

Presentation of the document

1.1 Aim

This document presents the installation, the configuration and the management of the components of the VISHNU toolkit.

1.2 Prerequisites

To ensure a good comfort during the reading of the document, the reader should have at least a basic experience in system administration on GNU/Linux environments. In particular and according to the operating system, a good knowledge of a package manager, such as *apt-get*, *dpkg*, *rpm*, *yum* or *zypper*, would be very helpful to ease the searching and the installation of VISHNU's dependencies.

Furthermore, we assume that you have read and understood the technical architecture of VISHNU. If not, refer to the document [ARCH], and especially in the chapter 4.

1.3 Structure of the document

This document contains the following parts:

- Definitions
- Installation
- Installation of the web services
- How to deploy
- Administration
- Reference of the commands
- Reference of the C++ API
- Reference of the Python API

Chapter 2

Definitions

2.1 Acronyms

- DB: Database, a centralized point providing mechanisms to store and retrieve application data in a effective way
- FMS: File Management Service.
- IMS: Information Management Service.
- LDAP: Lightweight Directory Access Protocol, is a protocol enabling to manage distributed directory services.
- MA: "Master Agent": Is a element of DIET allowing to distribute the requests between the clients and the servers.
- SQL: Language of request on the databases.
- TMS: Task Management Service.
- UMS: User Management Service
- WS: Web Services.

2.2 References

- [ARCH] D1.1g-VISHNU Technical Architecture: description of the architecture of the VISHNU application.
- [DIET_USERGUIDE] DIET User Guide: DIET is an open source tool used by VISHNU.
- [VISHNU_USERMANUAL] VISHNU User Manual.
- [VISHNU_API] VISHNU API: API VISHNU containing the signatures and the definition of the objects used by VISHNU.

2.3 Glossary

- FMS Client: It represents the programs allowing one to access the services of the FMS SeD.
- IMS Client: It represents the programs allowing one to access the services of the IMS SeD.
- TMS Client: It represents the programs allowing one to access the services of the TMS SeD.
- UMS Client: It represents the programs allowing one to access the services of the UMS SeD.
- gateway: It represents a machine sets before the gateway of the clusters.

- FMS SeD: It represents the program containing and executing the services of the FMS module.
- IMS SeD: It represents the program containing and executing the services of the IMS module.
- TMS SeD: It represents the program containing and executing the services of the TMS module.
- UMS SeD: It represents the program containing and executing the services of the UMS module.
- SysFera-DS: Is an open source middleware developed by SysFera. It is comprised of set of software that includes, among others, DIET and LOGSERVICE.

Chapter 3

Installing from the sources

3.1 Objectives

This chapter presents how to install VISHNU from the sources. It explains how to install both clients and servers (the installation of a server implies the installation of the associated client). The UMS module is necessary for all; any other module can be installed independantly from each other. It does not prevent using the services even thought servers are on another machine. Nevertheless, to enable IMS to load shed a machine, the corresponding modules (TMS and/or FMS) must be installed with IMS and running on the machine.

3.2 Prerequisite

According to the modules to install, you may need the following software dependencies:

- Dependency for all the modules VISHNU:
  - GCC: Version 4.4.3 or higher.
  - CMAKE: Version 2.8 or higher.
  - GNU Make or Ninja: Executable generators. Alternatively to Make, Ninja generator is more faster and is helpful for reducing the compilation duration. Especially when you need to compile several modules of VISHNU on the same machine.
  - BOOST: Version 1.46.1 or higher. At least the following modules *program_options, date_time, thread, filesystem, system, unit_test_framework, serialization, random* and *regex* should be installed.
  - OMNIORB: Version or 4.1.4 or higher.
  - GLIBC: Version 2.7 or higher.
  - OpenSSH: Version 4.2 or higher.
  - DIET: Version 2.8 with LOG SERVICE enabled.
  - Library to access the database:
    VISHNU requires a database; MySQL and PostGreSQL databases are supported. According to the type of the database engine, you need to install the appropriate API while taking care about the supported versions:
    * API PostGreSQL (PGSQL-API): Version 8.0 or higher.
    * API MySQL: Version 5.1 or higher.
- Optional dependencies for UMS:
  - OpenLDAP: Version 2.4 or higher.
- According to the underlying TMS Bacth Scheduler, you may need to install one of the following API:

  - Torque API: Tested with the version 2.3.6.
  - IBM LoadLeveler API: Tested with the version 3.5.1.14
  - SLURM API: Tested with the versions v2.2.x, 2.3.x and 2.4.x.
  - LSF API: Tested with the version v7.0.6.134609.
  - Grid Engine API: Tested with the version 2011.11.
- Optional dependency:
  - SWIG: 1.3.40. WARNING: The version 2 is not supported.
  - JAVA SDK: Version 1.6
  - Python: Version 2.x, x >= 5.

3.3 Compilation of the sources

VISHNU uses the CMake compilation system and follows the actual uses. The main options are:

- BUILD_TESTING, compile the tests (OFF by default).
- CLIENT_ONLY, do not compile the servers (OFF by defaut). Otherwize, both clients and servers are compiled.
- CMAKE_INSTALL_PREFIX, directory to install (/usr/local by default on the *nix plateform)
- COMPILE_TMS, compile the TMS module (OFF by default). If ON, COMPILE_UMS must be activated and the following options VISHNU_BATCH and VISHNU_BATCH_VERSION must be filled.
- COMPILE_IMS, compile the IMS module (OFF by default). If ON, COMPILE_UMS must be activated.
- COMPILE_FMS, compile the FMS module (OFF by default). If ON, COMPILE_UMS must be activated.
- COMPILE_UMS, compile the UMS module (ON by default).
- ENABLE_PYTHON, to compile the python API (OFF by default).
- ENABLE_JAVA, to compile the java for the WS API (OFF par default).
- VISHNU_USE_LDAP, to activate the LDAP support to authenticate (OFF by default).
- ENABLE_SWIG, to generate the code for the python and java modules (otherwise use a precompiled version if the flags are activated) (OFF by default). This option must be activated if some VISHNU modules are not compiled. It requires the swig package to compile (see dependencies).
- VISHNU_BATCH, compile the TMS SeD for the corresponding batch (TORQUE by default). The (BATCHNAME)_INCLUDE_DIR and (BATCHNAME)_LIB variables must be filled.
  - LOADLEVELER_INCLUDE_DIR indicate the absolute path to the llapi.h file
  - LOADLEVELER_LIB indicate the absolute path to the llapi.so library
  - TORQUE_INCLUDE_DIR indicate the absolute path to the pbs_ifl.h file.
  - TORQUE_LIB indicates the absolute path to the libtorque.so library.
  - SLURM_INCLUDE_DIR indicate the absolute path to the file slurm/slurm.h
  - SLURM_LIB indicate the absolute path to the libslurm.so library.
  - LSF_INCLUDE_DIR indicate the absolute path to the lsf/lsbatch.h file
  - LSBATCH_LIB indicate the absolute path to the libbat.so library.
  - LSF_LIB indicate the absolute path to the liblsf.so library
  - SGE_ROOT indicate the absolute path to the directory where Grid Engine is installed
  - SGE_INCLUDE_DIR indicate the absolute path to the drmaa.h file

  - SGE_BIN_DIR indique the absolute path to the Grid Engine binaries
  - SGE_LIB indicate the absolute path to the libdrmaa.so library
- VISHNU_BATCH_VERSION indique la version du batch scheduler utilisée

For instance, the following steps allow to compile and install the client and the server sides of UMS and TMS in /opt/vishnu. There, we use TORQUE as backend to TMS while enabling the Python API.

- 1. Create a build repository at the root of the vishnu project and go in

  $ *mkdir build*

  $ *cd build*
- 2. Generate the Makefile
  - Using Make

    $ *cmake -DCMAKE_INSTALL_PREFIX=/opt/vishnu* \
    -DENABLE_SWIG=ON \
    -DENABLE_PYTHON=ON \
    -DCOMPILE_UMS=ON \
    -DCOMPILE_TMS=ON \
    -DVISHNU_BATCH=TORQUE \
    -DVISHNU_BATCH_VERSION=2.3 \
    -DTORQUE_DIR=/opt/torque
  - Using Ninja: requires to set the generator to Ninja. See the flag CMAKE_GENERATOR.

    $ *cmake -DCMAKE_INSTALL_PREFIX=/opt/vishnu* \
    -DENABLE_SWIG=ON \
    -DENABLE_PYTHON=ON \
    -DCOMPILE_UMS=ON \
    -DCOMPILE_TMS=ON \
    -DVISHNU_BATCH=TORQUE \
    -DVISHNU_BATCH_VERSION=2.3 \
    -DTORQUE_DIR=/opt/torque \
    -DCMAKE_GENERATOR=Ninja
- 3. Launch the compilation
  - Using Make

    $ *make -j 2*

    The option -j 2 allows to run the compilation with two processes.
  - Using Ninja

    $ *ninja -j 2*

    The option -j 2 allows to run the compilation with two processes.
- 4. Installing the binaries

  $ *make install*

  This step may require root privileges.

Note: Do not forget to add the install directory/bin to PATH.

Chapter 4

Configuration of the database

The files to configure the DB are available in the core/database repository of the sources of VISHNU. Only postgresql and mysql are currently supported. Only one database is necessary for all the components of a VISHNU infrastructure (UMS, IMS, FMS and TMS).

4.1 Use a MySQL Database

This section assumes that you already have a MySQL installation ready to use. Otherwise refer to the MySQL documentation to learn how to install it.

You also need to have a root access on your MySQL installation.

The setup of the Vishnu database requires the scripts *mysql_create.sql* and *database_init.sql* located in *./core/database* from the source tree. The first script (mysql_create.sql) allows to create the database schema while the second one allows to carry out the first data initialization.

To create and initialize the database, perform the following steps:

- Connect to the MySQL server as root:

  $ *mysql -h mysql@server -u root -p* #replace mysql@server with the address of your MySql server
- Create the Vishnu's database

  $ *create database vishnu ;*
- Connect to the database

  $ *use vishnu ;*
- Create the database schema

  $ *source /path/to/mysql_create.sql*
- Initialize the database

  $ *source /path/to/database_init.sql*

4.2 Use a PostGreSQL Database

In this section, we assume that you have a PostGreSQL installation ready to use. Otherwise refer to the PostGreSQL documentation to learn how to install it.

You also need to have a root access on your PostGreSQL installation.

The setup of the Vishnu database requires the scripts *postgre_create.sql* and *database_init.sql* located in *./core/database* from the source tree. The first script (*postgre_create.sql*) allows to create the database schema while the second one allows to carry out the first data initialization.

To create and initialize the database, perform the following steps from the PostGreSQL server:

- Log as 'postgres' user (root):
  $ *su postgres*
- Create the Vishnu's database
  $ *createdb vishnu ;*
- Connect to the database
  $ *psql vishnu ;*
- Create the database schema
  $ \i */path/to/mysql_create.sql*
- Initialize the database
  $ \i */path/to/database_init.sql*

4.3 Use LDAP

We also assume here that you have a LDAP installation ready to use. Otherwise refer to the LDAP documentation to learn how to install it. On some GNU/Linux distribution, you may also install that from your package manager. On Debian systems, you need to install the following packages: *slapd, libldap-2.4-2, libldap2-dev* and *ldap-utils*.

To install with LDAP authentication support, the flag VISHNU_USE_LDAP must be enabled at the compilation time. Once the installation is completed, you need to set the parameter 'authenticationType' from the UMS configuration file accordingly. Possible values of this parameter are:

- UMS: Authenticate using only the database
- LDAPUMS: For each couple (username, password), try to authenticate using LDAP first then the UMS database
- UMSLDAP: For each couple (username, password), try to authenticate using the UMS database first then LDAP
- LDAP: Authenticate using only LDAP

Chapter 5

Installation of the web services

5.1 Prerequisite

- Install **Java 1.6** (command 'sudo apt-get install openjdk-6-jdk' or 'sudo apt-get install sun-java6-jdk' under Linux Debian) and check the value of the JAVA_HOME (set it if necessary).
- Install the desired modules (advice all with the flag CLIENT_ONLY)and the java options -DENABLE_JAVA=ON and -DENABLE_SWIG (**Note**: This prerequisite is optionnal because the jars are already available in the distribution.
- Install **Maven 2** to compile the jars (**Note**: This prerequisite is optional because the jars are already available in the distribution. Moreover Maven needs an internet connexion

5.2 Installation of JBoss

- Download the package JBOSSAS: (the binary version is available) at http://www.jboss.org/jbossas/downloads => version 5.1.0.GA
- Download the package JBOSSWS: (the binary version is available) at http://www.jboss.org/jbossws/downloads/ => version 3.3.1.GA
- Unzip the archive of the package JBOSSAS 5.1.0
- Define the environment variable JBOSS_HOME on the unzipped repository. For instance in the .bashrc file:
  'export JBOSS_HOME=/home/toto/jboss-5.1.0.GA'
- Unzip the archive of the package JBOSSWS 3.3.1 and go in the created repository
- Copy the file 'ant.properties.example' in 'ant.properties'
- Edit the newly created 'ant.properties' file. Set the value of the jboss510.home variable to the value JBOSS_HOME value. Note, if a version a jboss different to 5.1.0 is used, it is the corresponding variable that must be set.
- Launch the command 'ant deploy-jboss510'. Note, if a version a jboss different to 5.1.0 is used, it is the corresponding version that must be used in the command.
- The server can be launched running the script in JBOSS_HOME/bin/run.sh. To make the server available from an other machine, the '-b adressIP' option must be used. With adressIP representing the IP of the machien where the jboss server runs. Do not forget to check if the firewall enables the 8080 port.
- **Verification of the installation:** To check that the jboss server is running and that the WS is avaible, you can go to the URL 'http://localhost:8080/jbossws' (or 'http://adresseIP:8080/jbossws') and check that the page display the right version of jboss (jbossws-cxf-3.3.1.GA).

5.3 Installation of the WS modules with JBoss

5.3.1 Files to install

- VISHNULib-1.0-SNAPSHOT.jar
  - Contains the internal classes linking the JAVA (JNI) and the C++.
  - Copy it in **JBOSS_HOME/server/default/lib**.
  - Changing the jar implies restarting the jboss server.
  - Compilation (if necessary)
    - Go to the VISHNULib directory
    - Make 'mvn install' (may last long the first time)
    - The .jar file created is in the target directory
- WSAPI.jar
  - Contain the classes from the WSDL and the implementation of the WS.
  - Copy in **JBOSS_HOME/server/default/deploy**.
  - Can be updated without restarting the JBOSS server.
  - Compilation (if necessary)
    - Go in the WSAPI directoy after having a valid VISHNULib-1.0-SNAPSHOT.jar jar
    - Make 'mvn install' (may last long the first time)
    - The .jar file created is in the target directory.
    - Rename it to WSAP.jar and set it in JBOSS (necessary)
- libVISHNU.so
  - Necessary to use the WS in VISHNU. This library is obtained when compiling with the ENABLE_JAVA option.
  - It is installed in the lib directory with the other libs.
  - If there is a problem when deploying the JBOSS server, make sure it is located in the LD_LIBRARY_PATH.

5.3.2 Environment variable to be defined

- VISHNU_CONFIG_FILE: contains the full path to the client configuration file. Si the user guide [VISHNU_USER_GUIDE] for its content. If the execution fails with a message linked to the initialisation of the library, check this variable.
- LD_LIBRARY_PATH: contains the paths of the directories containing the VISHNU libraries, like libVISHNU.so.

5.3.3 Starting the JBoss server with the WS

- After installing as presented previously, the JBoss server must be started.
- **Verification of the installation:** To check that the JBoss server is correctly running and that the UMS web service is activated, go to the URL at 'http://localhost:8080/jbossws/services' (ou 'http://adresseIP:8080/jbossws/services') and check that the "service endPoint": **VISHNUUMSPortImpl** is active.

Chapter 6

Deploying VISHNU

6.1 On a single network

Modules can be deployed manually. Deployment will be described using the 4.1 figure of the [ARCH] document as an example. To simplify, we suppose the elements on the same area "dedicated to VISHNU" will be deployed on the same machine called **gateway**.

IMPORTANT NOTES:

- It is advised to avoid launching the same SeD on a machine more than once.
- There is a known bug under debian with boost file system, used by VISHNU. The report can be found there: https://svn.boost.org/trac/b
  If when launching a SeD the following error message appears: std::runtime_error: locale::facet::_S_create_c_locale name not valid, making "export LANG=C" should fix it.
- The VISHNU user 'root' must have a local account on each machine where there is an IMS server. The associated unix account will be used to relaunch the SeD.

Now that these points are clear, we can launch the environment. For starting your VISHNU environment successfully, you need to perform the following steps careful:

1. Check you have a database created and initialized.
2. Start the CORBA naming service on the gateway. The command is **omniNames -start** if starting for the first time, **omniNames** is enough otherwise. Warning, in the omniNames configuration file, use the address of the host and not 'localhost' or '127.0.0.1'.
3. Start the log service of DIET on the gateway. The command is **LogCentral --config config.cfg** with config.cfg a configuration file for the log central. An example of configuration is provided at the section 6.10.
4. Start the MA with its configuration file on the gateway: **dietAgent config.cfg**. See the paragraph 6.3 to have an example of configuration file. This configuration file can contain the 3 following lines:
   - 'traceLevel = 0': The level of verbosity of the agent, can be between 0 and 100. (100 is fully verbose, with corba details).
   - 'agentType = DIET_MASTER_AGENT': The type of the agent, the other value possible is DIET_LOCAL_AGENT, but in our case it must be DIET_MASTER_AGENT.
   - 'name = MA0': The name of the agent, for instance MA0 here.
5. Launch the UMS SeD on the gateway with the command **umssed ~/ums_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.5 to see an example of configuration file. The parameters correspond to:
   - 'dietConfigFile=/usr/local/etc/SeD.cfg': the path until the DIET configuration file. See the paragraph 6.4 for an example. The file can contains the two following lines:

  - 'traceLevel = 0': The level of verbosity of the agent, can be between 0 and 100. (100 is fully verbose, with corba details).
  - 'parentName = MA0': The name of the master agent to be linked to. It must be the exact same name as the one in the 'name' field in the corresponding MA. In this example, it is MA0.
  - 'useLogService = 1': To use the log service of DIET.
- 'vishnuId=1': The id of the VISHNU deployment to use in te DB (1 by default).
- 'databaseType=postgresql': To use a postgresql database. To use a MySQL database, one must set the value to 'mysql'.
- 'databaseHost=localhost': The DNS name of the DB server, or 'localhost' if the DB is local.
- 'databaseName=vishnu': The DB name.
- 'databaseUserName=vishnu_user': The username to connect to the DB.
- 'databaseUserPassword=vishnu_user': The password to connect to the DB.
- 'databaseConnectionsNb=5': The number of DB connexions that can be opened on the DB. By default it is 10.
- 'sendmailScriptPath=/usr/local/vishnu/sbin/sendmail.py': The script used to send mails. It is installed with the UMS module in the sbin directory.
- 'authenticationType=LDAP': Use only LDAP to authentify the users (see LDAP for the 4 values available).
- 'vishnuMachineId=machine_1' : Optionnal, this parameter specifies the identifier of the machine related to the SeD.
  IMPORTANT: During the first initialization, i.e. when there is not yet a machine registered into the system, this parameter must be leave empty. Indeed, when set, its value should correspond to a VISHNU machine, otherwise the SeD will fail.

6. On a torque machine, launch the torque server (pbs_serv), the torque scheduler (pbs_sched) and the torque scheduler (pbs_mom) of Torque.
7. Start the TMS SeD on the Torque host machine. The command is: **tmssed ~/tms_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.6 to have an example. The parameters are the same as the ones of UMS plus:
   - 'intervalMonitor = 1': The frequency (in second) when the database is updated with the jobs states.
   - 'batchSchedulerType=TORQUE': The type of batch scheduler used.
   - 'vishnuMachineId=machine_1': The VISHNU machine identifier where the TMS SeD is launched (the same as given by vishnu_list_machines).
8. On the SLURM machine, launch the servers *slurmd, slurmctld* and *slurmdbd*.
9. Start the TMS SeD on the Torque host machine. The command is: **tmssed ~/tms_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.6 to have an example. The parameters are the same as the ones of UMS plus:
   - 'intervalMonitor = 1': The frequency (in second) when the database is updated with the jobs states.
   - 'batchSchedulerType=SLURM': The type of batch scheduler used.
   - 'vishnuMachineId=machine_1': The VISHNU machine identifier where the TMS SeD is launched (the same as given by vishnu_list_machines).
10. On the LSF machine, launch the executables *hostsetup lsfstartup*.
11. Start the TMS SeD on the Torque host machine. The command is: **tmssed ~/tms_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.6 to have an example. The parameters are the same as the ones of UMS plus:
    - 'intervalMonitor = 1': The frequency (in second) when the database is updated with the jobs states.
    - 'batchSchedulerType=LSF': The type of batch scheduler used.
    - 'vishnuMachineId=machine_1': The VISHNU machine identifier where the TMS SeD is launched (the same as given by vishnu_list_machines).
12. On the Grid Engine machine, launch the executables *sge_qmaster* and *sge_execd*.
13. Start the TMS SeD on the Torque host machine. The command is: **tmssed ~/tms_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.6 to have an example. The parameters are the same as the ones of UMS plus:

- 'intervalMonitor = 1': The frequency (in second) when the database is updated with the jobs states.
- 'batchSchedulerType=SGE': The type of batch scheduler used.
- 'vishnuMachineId=machine_1': The VISHNU machine identifier where the TMS SeD is launched (the same as given by vishnu_list_machines).

14. Start the IMS SeD on all the desired machines. The command is: **imssed ~/ims_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.5 to have an example. The parameters are the same as the ones of UMS. Please note that an IMS SeD must be launched on each machine to monitor.
15. Start the FMS SeD on a machine. The command is: **fmssed ~/fms_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.7 to have an example. The parameters are the same as the ones of UMS plus:
    - 'intervalMonitor = 1': The frequency (in second) when the database is updated with the file transfers.
16. VISHNU modules are ready to be used. To do so, a client must connect and submit requests to VISHNU using the API of the modules (see the user guide for more information).

6.2 Deployment on various networks with DIET version >= 2.8

VISHNU modules can be deployed manually. The example of deployment will be based on figure 4.2 of the [ARCH] document, supposing to simplify that on a network, all the processes run on the same machine and that "network C" and "network D" are the same (no need of forwarder between them).

1. Check you have a database created and initialized.
2. Start the CORBA naming service on the gateway. The command is **omniNames -start** if starting for the first time, **omniNames** is enought otherwise. Warning, in the omninames configuration file, use the adress of the host and not 'localhost' or '127.0.0.1'.
3. Start a daemon forwarder on the network B:

   dietForwarder --name forwarder2

   - forwarder2: Name that identifies the forwarder
4. Start a daemon forwarder on the network C:

   dietForwarder --peer-name forwarder2 --ssh-host nom_DNS_machine_distante --remote-port 50005 --name forwarder --remote-host localhost --ssh-login login

   - --peer-name: The name of the peer forwarder, here forwarder2.
   - --ssh-host: The DNS name of the distant machine to connect the SSH tunnel.
   - --remote-port: optionnal, the port to use, here 50005.
   - --name: The name of the forwarder.
   - --remote-host: The loopback adresse, here 'localhost'.
   - --ssh-login: The login on the distant machine.
5. If the SSH key is accessible and that the key is protected by a passphrase, it is asked and the SSH connexion is established.
6. Start a daemon forwarder on the network B:

   logForwarder --name logforwarder2

   - logforwarder2: Name that identifies the forwarder
7. Start a daemon forwarder on the network C:

   logForwarder --peer-name logforwarder2 --ssh-host nom_DNS_machine_distante --remote-port 50005 --name logforwarder --remote-host localhost --ssh-login login

- --peer-name: The name of the peer forwarder, here forwarder2.
- --ssh-host: The DNS name of the distant machine to connect the SSH tunnel.
- --remote-port: optionnal, the port to use, here 50005.
- --name: The name of the forwarder.
- --remote-host: The loopback adresse, here 'localhost'.
- --ssh-login: The login on the distant machine.

8. If the SSH key is accessible and that the key is protected by a passphrase, it is asked and the SSH connexion is established.
9. Start the log service of DIET on the gateway. The command is **LogCentral --config config.cfg** with config.cfg a configuration file for the log central. See the paragraph 6.10 to have an example of configuration file.
10. Start the MA with its configuration file on the gateway: **dietAgent config.cfg**. See the paragraph 6.3 to have an example of configuration file. This configuration file can contain the 3 following lines:
    - 'traceLevel = 0': The level of verbosity of the agent, can be between 0 and 100. (100 is fully verbose, with corba details).
    - 'agentType = DIET_MASTER_AGENT': The type of the agent, the other value possible is DIET_LOCAL_AGENT, but in our case it must be DIET_MASTER_AGENT.
    - 'name = MA0': The name of the agent, for instance MA0 here.
11. Launch the UMS SeD on the network C with the command **umssed ~/ums_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.5 to see an example of configuration file. The parameters are the same as for deploying on a single network.
12. The UMS module is ready to be used. To do so, a client must connect et submit requests. If the client is on another network, other forwarders may be needed.

6.3 Deployment on various networks with DIET version 2.7

VISHNU modules can be deployed manually. The example of deployment will be based on figure 4.2 of the [ARCH] document, supposing to simplify that on a network, all the processes run on the same machine and that "network C" and "network D" are the same (no need of forwarder between them).

1. Having a database created and initialized.
2. Start the CORBA naming service on the gateway. The command is **omniNames -start** if starting for the first time, **omniNames** is enought otherwise. Warning, in the omninames configuration file, use the adress of the host and not 'localhost' or '127.0.0.1'.
3. Start a daemon forwarder on the network B:

   dietForwarder --name forwarder2 --net-config forwarder2.cfg

   - forwarder2: Name that identifies the forwarder
   - forwarder2.cfg: Configuration file (see the paragraph 6.8) containing the rules applied to the connexions between the networks. There are 2 rules: 'accept:' or 'reject:'. In the example of the paragraph 6.8, rules are:
     - a rule 'accept' corresponding to filters on the source of the accepted connexion. '.*' means all the connexions from all IP.
     - 2 rules to reject 2 addresses. It contains the local IP of the server.
4. Start a daemon forwarder on the network C:

   dietForwarder -C --peer-name forwarder2 --ssh-host nom_DNS_machine_distante --remote-port 50005 --name forwarder --remote-host localhost --net-config forwarder1.cfg --ssh-login login

   - -C: indicate it is the client.
   - --net-config: the configuration file. Contains rules similar to forwarder2.cfg

- --peer-name: The name of the peer forwarder, here forwarder2.
- --ssh-host: The DNS name of the distant machine to connect the SSH tunnel.
- --remote-port: optionnal, the port to use.
- --name: The name of the forwarder.
- --remote-host: The loopback adresse, here 'localhost'.
- --ssh-login: The login on the distant machine.

5. If the SSH key is accessible and that the key is protected by a passphrase, it is asked and the SSH connexion is established.
6. Start a daemon forwarder on the network B:

   logForwarder --name logforwarder2 --net-config logforwarder2

   - logforwarder2: Name that identifies the forwarder
   - logforwarder2.cfg: Configuration file (see the paragraph 6.8) containing the rules applied to the connexions between the networks. There are 2 rules: 'accept:' or 'reject:'. In the example of the paragraph 6.8, rules are:
     - a rule 'accept' corresponding to filters on the source of the accepted connexion. '.*' means all the connexions from all IP.
     - 2 rules to reject 2 addresses. It contains the local IP of the server.
7. Start a daemon log forwarder on the network C:

   logForwarder -C --peer-name logforwarder2 --ssh-host nom_DNS_machine_distante --remote-port 50005 --name logforwarder --remote-host localhost --net-config logforwarder1.cfg --ssh-login login

   - -C: indicate it is the client.
   - --peer-name: The name of the peer forwarder, here forwarder2.
   - --net-config: the configuration file. Contains rules similar to forwarder2.cfg
   - --ssh-host: The DNS name of the distant machine to connect the SSH tunnel.
   - --remote-port: optionnal, the port to use.
   - --name: The name of the forwarder.
   - --remote-host: The loopback adresse, here 'localhost'.
   - --ssh-login: The login on the distant machine.
8. If the SSH key is accessible and that the key is protected by a passphrase, it is asked and the SSH connexion is established.
9. Start the log service of DIET on the gateway. The command is **LogCentral --config config.cfg** with config.cfg a configuration file for the log central.
10. Start the MA with its configuration file on the gateway: **dietAgent config.cfg**. See the paragraph 6.3 to have an example of configuration file. This configuration file can contain the 3 following lines:
    - 'traceLevel = 0': The level of verbosity of the agent, can be between 0 and 100. (100 is fully verbose, with corba details).
    - 'agentType = DIET_MASTER_AGENT': The type of the agent, the other value possible is DIET_LOCAL_AGENT, but in our case it must be DIET_MASTER_AGENT.
    - 'name = MA0': The name of the agent, for instance MA0 here.
11. Launch the UMS SeD on the network C with the command **umssed ~/ums_sed.cfg**. The parameter is a VISHNU configuration file. See the paragraph 6.5 to see an example of configuration file. The parameters correspond to:
12. The UMS module is ready to be used. To do so, a client must connect et submit requests. If the client is on another network, other forwarders may be needed.

6.4 Example of a MA configuration file

```
traceLevel = 0
name = MA0
agentType = DIET_MASTER_AGENT
```

6.5 Example of a DIET SeD configuration file

```
traceLevel = 0
parentName = MA0
useLogService = 1
```

6.6 Example of a UMS configuration file

```
# Configuration of the VISHNU UMS SeD
dietConfigFile=/usr/local/etc/SeD.cfg
vishnuId=1
databaseType=postgresql
databaseHost=localhost
databaseName=vishnu
databaseUserName=vishnu_user
databaseUserPassword=vishnu_user
databaseConnectionsNb=5
sendmailScriptPath=/usr/local/sbin/sendmail.py
vishnuMachineId=machine_1
authenticationType=LDAP
```

6.7 Example of a TMS SeD configuration file

```
# Configuration of the VISHNU TMS SeD
dietConfigFile=/usr/local/etc/SeD.cfg
vishnuId=1
databaseType=postgresql
databaseHost=localhost
databaseName=vishnu
databaseUserName=vishnu_user
databaseUserPassword=vishnu_user
databaseConnectionsNb=5
sendmailScriptPath=/usr/local/sbin/sendmail.py
vishnuMachineId=machine_1
# TMS Configuration
# batchSchedulerType=SLURM if the batchScheduler is SLURM
# batchSchedulerType=LSF if the batchScheduler is LSF
# batchSchedulerType=SGE if the batchScheduler is SGE
# batchSchedulerType=LOADLEVELER if the batchScheduler is LL
batchSchedulerType=TORQUE
intervalMonitor = 1
```

6.8 Example of a FMS SeD configuration file

```
# Configuration of the VISHNU FMS SeD
dietConfigFile=/usr/local/etc/SeD.cfg
vishnuId=1
databaseType=postgresql
databaseHost=localhost
databaseName=vishnu
databaseUserName=vishnu_user
databaseUserPassword=vishnu_user
databaseConnectionsNb=5
vishnuMachineId=machine_1
# FMS Configuration
intervalMonitor = 1
```

6.9 Example of a forwarder configuration file

```
# accept everything from everyone
accept:.*

# reject its own ip
reject:192\.168\.1\.6
```

6.10 Example of a LogCentral configuration file

```
# an empty configuration file for LogService
[General]

[DynamicTagList]
[StaticTagList]
[UniqueTagList]
[VolatileTagList]
```

6.11 Configuration to send mails in VISHNU

The UMS SeD process uses the 'sendmail.py' file (in the sbin/ directory) to send mails to the users during some operations. This file can be updated by the administrator to configure it. By default, the configuration connects to the SMTP of 'localhost' using the port 25, without authentication.

The following parameters can be configured sendmail.py:

Option	Line of the sendmail.py script to modify
login	`parser.add_option("--login", dest="login", help="", default="[ ↩ login_utilisateur]")`
password	`parser.add_option("--password", dest="password", help="smtp password", ↩ default="[password_utilisateur]")`

Option	Line of the sendmail.py script to modify
hostname	`parser.add_option("--hostname", dest="host", help="smtp host", default="[ nom_serveur_SMTP]")`
port	`parser.add_option("--port", dest="port", help="smtp port [default: 25]", type=int, default="[no_port]")`
SSL	`parser.add_option("--ssl", action="store_true", dest="use_ssl", help="enable ssl support [default: %default - default port: 587 ]", default=True)`

6.12 Configuration of the private/public ssh keys for FMS

All the FMS commands are launched throught SSH under the user that has submitted the request name. FMS services are of 2 types: the ones using one remote machine and the ones using two machines.

- In the first case, the SeD connects on the distant machine and do the service. Consequently, the SeD's public key must be added to the user authorized_keys file ($HOME/.ssh/authorized_keys) of the corresponding distant machine.
- In the second case, 2 SSH connexions are necessary. The SeD connects to the source machine and starts the transfer (second connexion) to the distant machine. Consequently:
  - The SeD's public key must be added to the source machine authorized key file to enable the first connexion.
  - The source machine must be able to connect to the destination machine using ssh, using the private key registered in the VISHNU database added when creating the user's local account linking the machine to VISHNU. Besides, if the machine forward agent is activated between these machines, it is not necessary to use another ssh key couple between the source and destination machine.

To sum up, the SeD's SSH public key must be added on all the users' account of the machines used to fulfill the FMS services. All the keys, protected by passphrases must be stocked by an SSH agent to allow the automatic authentication.

6.13 Configuration of the private/public keys for TMS

The commands to submit, cancel and get the results of the jobs are executed throught SSH under the user name. To execute the commands, the TMS SeD public key's account must be added to the authorized_keys file ($HOME/.ssh/authorized_keys) of the user. All passphrase protected keys must be stocked by an SSH agent to allow the automatic authentication.

6.14 Test to execute a client command throught the shell API

1. Once the platform is installed and deployed, go to a client machine with VISHNU client installed. See the VISHNU user guide concerning this installation.
2. Export the environment variable VISHNU_CONFIG_FILE to a client configuration file (see the VISHNU user guide again for more).
3. Open a VISHNU session

   #vishnu_connect -u user

   Replace 'user' by a real user identifier. By default, VISHNU comes with a user named 'root' having all rights on the system (ID: 'root', Password: 'vishnu_user').

4. On the client, there should be a display of a session identifier, proving that the connexion was a success. On the MA and the SeD, depending on the verbosity level, there may be some information displayed. The client display should be similar to: **sessionId: root-2011-Jul-11-14:22:14.403491:86690**.

5. Close the session

   #*vishnu_close*

   No error should be raised.

Chapter 7

Administration

7.1 Presentation

The UMS module corresponds to user and machine handling in VISHNU. It allows to save a VISHNU configuration and restore it if needed. In this chapter, we will suppose that the user is connected as a VISHNU administrator. The commands presented are the shell one's, but all is the same throught the other API (python, WS, C++).

The whole API is available in the [VISHNU_API] document

7.2 User handling (UMS)

1. Adding a user is made with vishnu_add_user. It takes as a parameter the user name, it's familly name, it's accreditation level in VISHNU (user or admin) and its mail address. All these are mandatory. An accreditation level of 1 means admin, 0 means simple user. The userid is generated and returned.
2. Updating a user must be done by an admin. It is made throught the 'vishnu_update_user' command and enables to update a user first and last name, its status (can be locked), its mail address. The generated userid must be used to indicate the user to update.

   Note: Setting a user status to 'INACTIVE' corresponds to locking its VISHNU account, he cannot connect anymore.
3. Deleting a user delete its data from the database and is made with the vishnu_delete_user command.
4. Listing the users can be made by an admin only. The command is 'vishnu_list_users' and can take options like a specific userid.
5. Only an admin can reinitialize a user password in VISHNU. The command is 'vishnu_reset_password' with the userid of the user to change its password. At the following connexion, the user will be asked to change it's password (using the 'vishnu_change_password' command).

7.3 Handling machines (UMS+IMS)

1. Adding a machine is made with the 'vishnu_add_machine' command. It takes as a parameter the machine name, its location, the language of the given machine description and the SSH public key. These parameters are mandatory. Once the machine is added, its identifier is returned.
2. A machine is updated with the 'vishnu_update_machine'. The machine identifier is needed to update.

   Note: Changing a machine status to 'INACTIVE' corresponds to blocking it, the machine is unavailable to VISHNU users', but the admin can still see it.

3. Deleting a machine is made with the 'vishnu_delete_machine', with the machine identifier returned at the machine creation. It deletes from the database with all the asociated data. Warning it is not reversible.
4. Users can list machines, but the admin have another option, a userid, to list the machines where the given user as a VISHNU account.
5. Updating a machine technical description is made with the 'vishnu_set_system_info' command. The machine identifier is required.

7.4 Handling the plateform (UMS)

1. The admin can save at a given time VISHNU state's. It saves the users, machines, users' account, authentication systems. The function is 'vishnu_save_configuration' and does not need parameters and generates a save file.
2. An admin can load a saved configuration of VISHNU using the 'vishnu_restore_configuration' command that takes the saved file. To start this command, all the VISHNU user's must de disconnected.
3. An admin can define user's option default values for all the users (these options are the default time before the session is automatically closed, how the session closes: timeout or wait for an explicit close call, default copy tool: scp or rsync). The function is 'vishnu_configure_default_option' with the option name and its new default value.
4. An admin can add or uptade the authentication systems. For instance, it can add various LDAP to authenticate its users. Currently, for the same LDAP, if the users are in different branches, we need to add twice the LDAP with different names, but it allows the user to only give its LDAP login and not its branch. It is the admin, when he creates the authentication system, that defines the branch to use in the DAP tree. When the admin fills the path in the LDAP tree, he must replace the username by the \$USERNAME string, this string will be replaced by the actual user name. associated functions: vishnu_add_auth_system, vishnu_update_auth_system, vishnu_delete_auth_system

7.5 Admin options in user functions(UMS+FMS)

1. In 'vishnu_connect', an admin can give a VISHNU user identifier to connect as this user in VISHNU.
2. In 'vishnu_list_history_cmd', an admin can list all the commands of all the users or the commands of a particular user giving its identifier.
3. In 'vishnu_list_local_accounts', an admin can list all the accounts of all the users or the accounts of a particular user.
4. In 'vishnu_list_options', an admin can list all the options of all the users or the options of a particular user.
5. In 'vishnu_list_sessions', an admin can list all the sessions of all the users or the sessions of a user, filtering by a machine.
6. In 'vishnu_list_file_transfers', an admin can list all the file transfers of all the users, or the transfers of a particular user, or the transfers on a machine (may be source or destination)
7. In 'vishnu_stop_file_transfers', an admin can stop the file transfers of all the users, or a particular user, or on a machine

7.6 Handling processes in VISHNU and load shedding(IMS)

- An admin can list all the running VISHNU processes, on all the platform or on a particular machine. associated function: vishnu_get_processes.
- An admin can stop a VISHNU process, it will not be automatically restarted. Warning, the admin must have a local account on the machine. associated function: vishnu_stop
- An admin can restart a VISHNU process on a machine, this process must have already runned and he must have an account on the machine. associated function: vishnu_restart
- An admin can load shed a machine using two modes. In the HARD one, all the VISHNU processes are stopped on the machine, in the SOFT, FMS and TMS have their jobs and file transfers cancelled. associated function: vishnu_load_shed

7.7 Monitoring the machines (IMS)

- An admin can fix the frequency the machine state is recorded. associated function: vishnu_set_update_frequency
- An admin can get the frequenc the machine state is saved. associated function: vishnu_get_update_frequency
- An admin can fix a threshold on a machine. It can be the CPU use, the free remaining ram and the free remaining diskspace. When recording a machine state, if a threshold is reached, a mail is automatically sent to an admin responsible for the threshold on the machine. associated function: vishnu_set_threshold
- An admin can get the defined thresholds. associated function: vishnu_get_threshold

7.8 Defining the format of the identifiers (IMS)

- An admin can fix the format of the identifiers automatically generated for the users, machines, jobs, file transfers and authentication systems. These identifiers can contain various variables:
  - '$DAY': Variable replaced by the day (1-31)
  - '$MONTH': Variable replaced by the month (1-12)
  - '$YEAR': Variable replaced by the year (0-99)
  - '$CPT': Variable replaced by a counter value, to guarantee the unicity
  - '$SITE': For users and machines, replaced by the place
  - '$UNAME': For users, replaced by their first name
  - '$MANAME': For machines, replace by their name

  Warning, the counter variable is mandatory to avoid generating twice the same identifier. associated function: define_file_format, define_machine_format, define_task_format, define_user_format, define_auth_format.

Chapter 8

UMS Command reference

8.1 vishnu_add_user

vishnu_add_user — adds a new VISHNU user

Synopsis

`vishnu_add_user` [`-h`] *`firstname lastname privilege email`*

DESCRIPTION

This command allows an admin to add a new user in VISHNU. Several user information are necessary such as: lastname, firtsname and email address. The admin also gives a VISHNU privilege to the new user and a new userId and password are sent to the user by email.

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 already exists in the database" [22]

"The user is locked" [23]

"The user is not an administrator" [25]

"The mail adress is invalid" [27]

"The session key is unrecognized" [28]

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

"The machine is locked" [34]

EXAMPLE

To add the user Jean DUPONT as a simple user and with the mail dupont@gmail.com:

vishnu_add_user Jean DUPONT 0 dupont@gmail.com

8.2 vishnu_update_user

vishnu_update_user — updates the user information except the userId and the password

Synopsis

`vishnu_update_user [-h] [-f firstname] [-l lastname] [-p privilege] [-m email] [-s status] userId`

DESCRIPTION

This command allows an admin to update a VISHNU user information or to lock a user. When a user is locked, she/he can not uses VISHNU. However, it is not possible to change the privilege of another admin.

OPTIONS

-h *help* help about the command.

-f *firstname* represents the updated firstname of the user.

-l *lastname* represents the updated lastname of the user.

-p *privilege* represents the updated privilege of the user. The value must be an integer. Predefined values are: 0 (USER), 1 (ADMIN).

-m *email* represents the updated email address of the user.

-s *status* represents the status of the user (LOCKED or ACTIVE). The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (INACTIVE), 1 (ACTIVE).

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]

"Trying to lock a user account that is already locked" [24]

"The user is not an administrator" [25]

"The mail adress is invalid" [27]

"The session key is unrecognized" [28]

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

EXAMPLE

To update the mail address of a user user_1 to jdupont@gmail.com:

vishnu_update_user -m jdupont@gmail.com user_1

8.3 vishnu_delete_user

vishnu_delete_user — removes a user from VISHNU

Synopsis

`vishnu_delete_user` [`-h`] *`userId`*

DESCRIPTION

This command allows an admin to delete a user from VISHNU. When a user is deleted from VISHNU all of her/his information are deleted from VISHNU. However, it is not possible to delete the VISHNU root user.

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 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]

EXAMPLE

Tu delete the user user_1:

vishnu_delete_user user_1

8.4 vishnu_reset_password

vishnu_reset_password — resets the password of a user

Synopsis

`vishnu_reset_password` [-h] *userId*

DESCRIPTION

This command allows an admin to reset the password of the user. The password generated is temporary and must be changed for using 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 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]

EXAMPLE

To reset the password of the user user_1:

vishnu_reset_password user_1

8.5 vishnu_save_configuration

vishnu_save_configuration — saves the configuration of VISHNU

Synopsis

`vishnu_save_configuration [-h]`

DESCRIPTION

This commands allows an admin to save the VISHNU configuration. This configuration contains the list of users, the lists of machines and the list of local user configurations. It is saved on a xml format on a file registered on the directory $HOME/.vishnu/configurati

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 user is not an administrator" [25]

"A problem occurs during the configuration saving " [39]

EXAMPLE

To save the current configuration:

vishnu_save_configuration

8.6 vishnu_restore_configuration

vishnu_restore_configuration — restores the configuration of VISHNU

Synopsis

`vishnu_restore_configuration` [`-h`] *`filePath`*

DESCRIPTION

This function must be used carefully as it replaces all the content of the VISHNU central database with the information stored in the provided file. This file contains the list of users, the lists of machines and the list of local user configurations. It can be created using the vishnu_save_configuration command. The "root" VISHNU user is the only user authorized to call this function, and this action must be done without any other user connected to VISHNU. After restoring, the vishnu database is re-initialized.

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 user is not an administrator" [25]

"A problem occurs during the configuration restoring" [40]

EXAMPLE

To restore the configuration in /tmp/toto.cfg:

vishnu_restore_configuration /tmp/toto.cfg

8.7 vishnu_add_machine

vishnu_add_machine — adds a new machine in VISHNU

Synopsis

`vishnu_add_machine [-h] name site language sshPublicKeyFile machineDescription`

DESCRIPTION

This command allows an admin to add a new machine in VISHNU. Several machine information are mandatory such as: name, site, language and the public ssh key of the VISHNU system account on the machine. This public key will be provided automatically to all new VISHNU users who will have to add it to the authorized keys of their own account on the machine.

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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machineId already exists in the database" [33]

"The closure policy is unknown" [42]

EXAMPLE

To add the machine perceval in paris with the description in french "ceci est une description" with the public key in /tmp/key.pub:

vishnu_add_machine perceval paris fr /tmp/key.pub "ceci est une description"

8.8 vishnu_update_machine

vishnu_update_machine — updates machine description

Synopsis

`vishnu_update_machine [-h] [-n name] [-s site] [-d machineDescription] [-l language] [-t status] [-k sshPublicKeyFile] machineId`

DESCRIPTION

This command allows an admin to update a VISHNU machine or to locked it. A machine locked is not usable.

OPTIONS

-h *help* help about the command.

-n *name* represents the name of the machine.

-s *site* represents the location of the machine.

-d *machineDescription* represents the description of the machine.

-l *language* represents the language used for the description of the machine.

-t *status* represents the status of the machine. The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (INACTIVE), 1 (ACTIVE).

-k *sshPublicKeyFile* contains the path to the SSH public key used by VISHNU to access local user accounts.

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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machine id is unknown" [32]

"The closure policy is unknown" [42]

EXAMPLE

To change the name of the machine whose id is machine_1 to provencal:

vishnu_update_machine machine_1 -n provencal

8.9 vishnu_delete_machine

vishnu_delete_machine — removes a machine from VISHNU

Synopsis

`vishnu_delete_machine` [`-h`] *`machineId`*

DESCRIPTION

This command allows an admin to delete a machine from VISHNU. When a machine is deleted all of its information are 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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machine id is unknown" [32]

EXAMPLE

To delete the machine machine_1:

vishnu_delete_machine machine_1

8.10 vishnu_list_users

vishnu_list_users — lists VISHNU users

Synopsis

`vishnu_list_users` [`-h`] [`-u` *`userId`*] [`-i` *`authSystemId`*]

DESCRIPTION

This command allows an admin to display all users information except the passwords.

OPTIONS

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

`-u` ***`userId`*** allows an admin to get information about a specific user identified by his/her userId.

`-i` ***`authSystemId`*** is an option to list users who have local user-authentication configurations on 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]

EXAMPLE

To list all the users:

vishnu_list_users

8.11 vishnu_configure_default_option

vishnu_configure_default_option — configures a default option value

Synopsis

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

DESCRIPTION

Options in VISHNU corresponds to 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 allows an administrator to configure the default value of an option; this is the value that will be applied when the user has no configuration defined for that option using the vishnu_configure_option 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 (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 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]

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

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

EXAMPLE

To configure the option VISHNU_TIMEOUT with the value 42:

vishnu_configure_default_option VISHNU_TIMEOUT 42

8.12 vishnu_add_auth_system

vishnu_add_auth_system — adds a new user-authentication system in VISHNU

Synopsis

`vishnu_add_auth_system` [`-h`] [`-b` *`ldapBase`*] *`name URI authLogin authPassword userPasswordEncryption type`*

DESCRIPTION

This command allows an admin to add a new user-authentication system in VISHNU. Several user-authentication system's information are mandatory such as: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

OPTIONS

-h *help* help about the command.

-b *ldapBase* is an option for user-authentication system based on LDAP which specifies the DN of the root entry .

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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The identifier (name or generated identifier) of the user-authentication system already exists" [50]

"The encryption method is unknown" [53]

EXAMPLE

To add an LDAP's user-authentication system on VISHNU named CLAMART with the parameters which follows: URI ldap://127.0.0.1:3
, login cn=ldapadmin,dc=sysfera,dc=fr, password secret and DN root entry dc=sysfera,dc=fr:

vishnu_add_auth_system CLAMART ldap://127.0.0.1:389/ cn=ldapadmin,dc=sysfera,dc=fr secret -b uid=\$USERNAME,ou=users,dc=

8.13 vishnu_update_auth_system

vishnu_update_auth_system — updates a user-authentication system in VISHNU

Synopsis

`vishnu_update_auth_system` [`-h`] [`-n` *name*] [`-i` *URI*] [`-u` *authLogin*] [`-w` *authPassword*] [`-e` *userPassw-ordEncryption*] [`-t` *type*] [`-s` *status*] [`-b` *ldapBase*] *authSystemId*

DESCRIPTION

This command allows an admin to update a user-authentication system in VISHNU. It is possible to change the parameters which follow: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

OPTIONS

-h *help* help about the command.

-n *name* corresponds to the user-authentication system's name.

-i *URI* is the URI of the user-authentication systems (by the form host:port for LDAP).

-u *authLogin* is the login used to connect to the user-authentication system.

-w *authPassword* is the password used to connect to the user-authentication system.

-e *userPasswordEncryption* represents the encryption method used to encrypt user's password. The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (SSHA).

-t *type* represents the type of the user-authentication system. The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (LDAP).

-s *status* represents the status of the user-authentication system. The value must be an integer. Predefined values are: -1 (UNDEFINED), 0 (INACTIVE), 1 (ACTIVE).

-b *ldapBase* is an option for user-authentication system based on LDAP which specifies the DN of the root entry .

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 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 system is already locked" [49]

"The encryption method is unknown" [53]

EXAMPLE

To change the address of a user-authentication system whose identifier is AUTHENLDAP_1:

vishnu_update_auth_system -i ldap://192.128.1.1:389/ AUTHENLDAP_1

8.14 vishnu_delete_auth_system

vishnu_delete_auth_system — removes a user-authentication system from VISHNU

Synopsis

`vishnu_delete_auth_system` [`-h`] *`authSystemId`*

DESCRIPTION

This command allows an admin to remove a user-authentication system 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 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]

EXAMPLE

To remove a user-authentication system whose identifier is AUTHENLDAP_1:

vishnu_delete_auth_system AUTHENLDAP_1

Chapter 9

UMS C++ API Reference

9.1 addUser

addUser — adds a new VISHNU user

Synopsis

int **vishnu::addUser**(const string& sessionKey, User& newUser);

DESCRIPTION

This command allows an admin to add a new user in VISHNU. Several user information are necessary such as: lastname, firtsname and email address. The admin also gives a VISHNU privilege to the new user and a new userId and password are sent to the user by email.

ARGUMENTS

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

newUser Input/Output argument. Object containing the new user 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 userId already exists in the database" [22]

"The user is locked" [23]

"The user is not an administrator" [25]

"The mail adress is invalid" [27]

"The session key is unrecognized" [28]

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

"The machine is locked" [34]

9.2 updateUser

updateUser — updates the user information except the userId and the password

Synopsis

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

DESCRIPTION

This command allows an admin to update a VISHNU user information or to lock a user. When a user is locked, she/he can not uses VISHNU. However, it is not possible to change the privilege of another admin.

ARGUMENTS

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

user Input argument. Object containing user 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 userId is unknown" [21]

"The user is locked" [23]

"Trying to lock a user account that is already locked" [24]

"The user is not an administrator" [25]

"The mail adress is invalid" [27]

"The session key is unrecognized" [28]

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

9.3 deleteUser

deleteUser — removes a user from VISHNU

Synopsis

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

DESCRIPTION

This command allows an admin to delete a user from VISHNU. When a user is deleted from VISHNU all of her/his information are deleted from VISHNU. However, it is not possible to delete the VISHNU root user.

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 who will be deleted from 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]

9.4 resetPassword

resetPassword — resets the password of a user

Synopsis

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

DESCRIPTION

This command allows an admin to reset the password of the user. The password generated is temporary and must be changed for using 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 password will be reset.

tmpPassword Output argument. The temporary password 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 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]

9.5 saveConfiguration

saveConfiguration — saves the configuration of VISHNU

Synopsis

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

DESCRIPTION

This commands allows an admin to save the VISHNU configuration. This configuration contains the list of users, the lists of machines and the list of local user configurations. It is saved on a xml format on a file registered on the directory $HOME/.vishnu/configurati

ARGUMENTS

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

configuration Output argument. The configuration is an object which encapsulates the configuration description.

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 user is not an administrator" [25]

"A problem occurs during the configuration saving " [39]

9.6 restoreConfiguration

restoreConfiguration — restores the configuration of VISHNU

Synopsis

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

DESCRIPTION

This function must be used carefully as it replaces all the content of the VISHNU central database with the information stored in the provided file. This file contains the list of users, the lists of machines and the list of local user configurations. It can be created using the vishnu_save_configuration command. The "root" VISHNU user is the only user authorized to call this function, and this action must be done without any other user connected to VISHNU. After restoring, the vishnu database is re-initialized.

ARGUMENTS

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

filePath Input argument. The filePath is the path of the file used to restore VISHNU configuration.

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 user is not an administrator" [25]

"A problem occurs during the configuration restoring" [40]

9.7 addMachine

addMachine — adds a new machine in VISHNU

Synopsis

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

DESCRIPTION

This command allows an admin to add a new machine in VISHNU. Several machine information are mandatory such as: name, site, language and the public ssh key of the VISHNU system account on the machine. This public key will be provided automatically to all new VISHNU users who will have to add it to the authorized keys of their own account on the machine.

ARGUMENTS

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

newMachine Input/Output argument. Is an object which encapsulates the information of the machine which will be added in VISHNU except the machine id which will be created automatically by VISHNU.

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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machineId already exists in the database" [33]

"The closure policy is unknown" [42]

9.8 updateMachine

updateMachine — updates machine description

Synopsis

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

DESCRIPTION

This command allows an admin to update a VISHNU machine or to locked it. A machine locked is not usable.

ARGUMENTS

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

machine Input argument. Existing machine 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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machine id is unknown" [32]

"The closure policy is unknown" [42]

9.9 deleteMachine

deleteMachine — removes a machine from VISHNU

Synopsis

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

DESCRIPTION

This command allows an admin to delete a machine from VISHNU. When a machine is deleted all of its information are deleted from VISHNU.

ARGUMENTS

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

machineId Input argument. MachineId represents the identifier of the 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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The machine id is unknown" [32]

9.10 listUsers

listUsers — lists VISHNU users

Synopsis

int **vishnu::listUsers**(const string& sessionKey, ListUsers& listuser, const ListUsersOptions& options = ListUsersOptions());

DESCRIPTION

This command allows an admin to display all users information except the passwords.

ARGUMENTS

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

listuser Output argument. Listuser is the list of users .

options Input argument. Allows an admin to get information about a specific user identified by his/her userId or to get information about users authenticated by 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 not an administrator" [25]

"The session key is unrecognized" [28]

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

9.11 configureDefaultOption

configureDefaultOption — configures a default option value

Synopsis

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

DESCRIPTION

Options in VISHNU corresponds to 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 allows an administrator to configure the default value of an option; this is the value that will be applied when the user has no configuration defined for that option using the vishnu_configure_option command.

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 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]

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

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

9.12 addAuthSystem

addAuthSystem — adds a new user-authentication system in VISHNU

Synopsis

int **vishnu::addAuthSystem**(const string& sessionKey, AuthSystem& newAuthSys);

DESCRIPTION

This command allows an admin to add a new user-authentication system in VISHNU. Several user-authentication system's information are mandatory such as: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

ARGUMENTS

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

newAuthSys Input/Output argument. Is an object which encapsulates the information of the user-authentication system 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 user is not an administrator" [25]

"The session key is unrecognized" [28]

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

"The identifier (name or generated identifier) of the user-authentication system already exists" [50]

"The encryption method is unknown" [53]

9.13 updateAuthSystem

updateAuthSystem — updates a user-authentication system in VISHNU

Synopsis

int **vishnu::updateAuthSystem**(const string& sessionKey, const AuthSystem& AuthSys);

DESCRIPTION

This command allows an admin to update a user-authentication system in VISHNU. It is possible to change the parameters which follow: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

ARGUMENTS

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

AuthSys Input argument. Is an object which encapsulates the information of the user-authentication system 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 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 system is already locked" [49]

"The encryption method is unknown" [53]

9.14 deleteAuthSystem

deleteAuthSystem — removes a user-authentication system from VISHNU

Synopsis

int **vishnu::deleteAuthSystem**(const string& sessionKey, const string& authSystemId);

DESCRIPTION

This command allows an admin to remove a user-authentication system from VISHNU.

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.

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 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]

Chapter 10

UMS Python API Reference

10.1 VISHNU.addUser

VISHNU.addUser — adds a new VISHNU user

Synopsis

ret=VISHNU.addUser(string sessionKey, User newUser);

DESCRIPTION

This command allows an admin to add a new user in VISHNU. Several user information are necessary such as: lastname, firtsname and email address. The admin also gives a VISHNU privilege to the new user and a new userId and password are sent to the user by email.

ARGUMENTS

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

newUser Input/Output argument. Object containing the new user 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 userId already exists in the database" [22])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The mail adress is invalid" [27])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

UMSVishnuException("The machine is locked" [34])

10.2 VISHNU.updateUser

VISHNU.updateUser — updates the user information except the userId and the password

Synopsis

ret=VISHNU.updateUser(string sessionKey, User user);

DESCRIPTION

This command allows an admin to update a VISHNU user information or to lock a user. When a user is locked, she/he can not uses VISHNU. However, it is not possible to change the privilege of another admin.

ARGUMENTS

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

user Input argument. Object containing user 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 userId is unknown" [21])

UMSVishnuException("The user is locked" [23])

UMSVishnuException("Trying to lock a user account that is already locked" [24])

UMSVishnuException("The user is not an administrator" [25])

UMSVishnuException("The mail adress is invalid" [27])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

10.3 VISHNU.deleteUser

VISHNU.deleteUser — removes a user from VISHNU

Synopsis

ret=VISHNU.deleteUser(string sessionKey, string userId);

DESCRIPTION

This command allows an admin to delete a user from VISHNU. When a user is deleted from VISHNU all of her/his information are deleted from VISHNU. However, it is not possible to delete the VISHNU root user.

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 who will be deleted from 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])

10.4 VISHNU.resetPassword

VISHNU.resetPassword — resets the password of a user

Synopsis

ret, tmpPassword=VISHNU.resetPassword(string sessionKey, string userId);

DESCRIPTION

This command allows an admin to reset the password of the user. The password generated is temporary and must be changed for using 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 password will be reset.

tmpPassword Output argument. The temporary password generated by VISHNU.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

tmpPassword(string) The temporary password generated by VISHNU

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])

10.5 VISHNU.saveConfiguration

VISHNU.saveConfiguration — saves the configuration of VISHNU

Synopsis

ret=VISHNU.saveConfiguration(string sessionKey, Configuration configuration);

DESCRIPTION

This commands allows an admin to save the VISHNU configuration. This configuration contains the list of users, the lists of machines and the list of local user configurations. It is saved on a xml format on a file registered on the directory $HOME/.vishnu/configurati

ARGUMENTS

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

configuration Output argument. The configuration is an object which encapsulates the configuration description.

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 user is not an administrator" [25])

UMSVishnuException("A problem occurs during the configuration saving " [39])

10.6 VISHNU.restoreConfiguration

VISHNU.restoreConfiguration — restores the configuration of VISHNU

Synopsis

ret=VISHNU.restoreConfiguration(string sessionKey, string filePath);

DESCRIPTION

This function must be used carefully as it replaces all the content of the VISHNU central database with the information stored in the provided file. This file contains the list of users, the lists of machines and the list of local user configurations. It can be created using the vishnu_save_configuration command. The "root" VISHNU user is the only user authorized to call this function, and this action must be done without any other user connected to VISHNU. After restoring, the vishnu database is re-initialized.

ARGUMENTS

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

filePath Input argument. The filePath is the path of the file used to restore VISHNU configuration.

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 user is not an administrator" [25])

UMSVishnuException("A problem occurs during the configuration restoring" [40])

10.7 VISHNU.addMachine

VISHNU.addMachine — adds a new machine in VISHNU

Synopsis

ret=VISHNU.addMachine(string sessionKey, Machine newMachine);

DESCRIPTION

This command allows an admin to add a new machine in VISHNU. Several machine information are mandatory such as: name, site, language and the public ssh key of the VISHNU system account on the machine. This public key will be provided automatically to all new VISHNU users who will have to add it to the authorized keys of their own account on the machine.

ARGUMENTS

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

newMachine Input/Output argument. Is an object which encapsulates the information of the machine which will be added in VISHNU except the machine id which will be created automatically by VISHNU.

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 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 machineId already exists in the database" [33])

UMSVishnuException("The closure policy is unknown" [42])

10.8 VISHNU.updateMachine

VISHNU.updateMachine — updates machine description

Synopsis

ret=VISHNU.updateMachine(string sessionKey, Machine machine);

DESCRIPTION

This command allows an admin to update a VISHNU machine or to locked it. A machine locked is not usable.

ARGUMENTS

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

machine Input argument. Existing machine 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 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 machine id is unknown" [32])

UMSVishnuException("The closure policy is unknown" [42])

10.9 VISHNU.deleteMachine

VISHNU.deleteMachine — removes a machine from VISHNU

Synopsis

ret=VISHNU.deleteMachine(string sessionKey, string machineId);

DESCRIPTION

This command allows an admin to delete a machine from VISHNU. When a machine is deleted all of its information are deleted from VISHNU.

ARGUMENTS

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

machineId Input argument. MachineId represents the identifier of the 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 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 machine id is unknown" [32])

10.10 VISHNU.listUsers

VISHNU.listUsers — lists VISHNU users

Synopsis

ret, listuser=VISHNU.listUsers(string sessionKey, ListUsersOptions options = ListUsersOptions());

DESCRIPTION

This command allows an admin to display all users information except the passwords.

ARGUMENTS

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

listuser Output argument. Listuser is the list of users .

options Input argument. Allows an admin to get information about a specific user identified by his/her userId or to get information about users authenticated by 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 not an administrator" [25])

UMSVishnuException("The session key is unrecognized" [28])

UMSVishnuException("The sessionKey is expired. The session is closed." [29])

10.11 VISHNU.configureDefaultOption

VISHNU.configureDefaultOption — configures a default option value

Synopsis

ret=VISHNU.configureDefaultOption(string sessionKey, OptionValue optionValue);

DESCRIPTION

Options in VISHNU corresponds to 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 allows an administrator to configure the default value of an option; this is the value that will be applied when the user has no configuration defined for that option using the vishnu_configure_option command.

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 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])

UMSVishnuException("The value of the timeout is incorrect" [43])

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

10.12 VISHNU.addAuthSystem

VISHNU.addAuthSystem — adds a new user-authentication system in VISHNU

Synopsis

ret=VISHNU.addAuthSystem(string sessionKey, AuthSystem newAuthSys);

DESCRIPTION

This command allows an admin to add a new user-authentication system in VISHNU. Several user-authentication system's information are mandatory such as: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

ARGUMENTS

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

newAuthSys Input/Output argument. Is an object which encapsulates the information of the user-authentication system 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 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 identifier (name or generated identifier) of the user-authentication system already exists" [50])

UMSVishnuException("The encryption method is unknown" [53])

10.13 VISHNU.updateAuthSystem

VISHNU.updateAuthSystem — updates a user-authentication system in VISHNU

Synopsis

ret=VISHNU.updateAuthSystem(string sessionKey, AuthSystem AuthSys);

DESCRIPTION

This command allows an admin to update a user-authentication system in VISHNU. It is possible to change the parameters which follow: URI, login, password, type and optionally for LDAP the DN of the root entry. By default, the type of the user-authentication system is LDAP.

ARGUMENTS

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

AuthSys Input argument. Is an object which encapsulates the information of the user-authentication system 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 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 system is already locked" [49])

UMSVishnuException("The encryption method is unknown" [53])

10.14 VISHNU.deleteAuthSystem

VISHNU.deleteAuthSystem — removes a user-authentication system from VISHNU

Synopsis

ret=VISHNU.deleteAuthSystem(string sessionKey, string authSystemId);

DESCRIPTION

This command allows an admin to remove a user-authentication system from VISHNU.

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.

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 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])

Chapter 11

IMS Command reference

11.1 vishnu_get_processes

vishnu_get_processes — displays the list of the VISHNU processes running on machines

Synopsis

`vishnu_get_processes` [`-h`] [`-p` *`machineId`*]

DESCRIPTION

This command with restricted access is used to get the list of VISHNU server processes that are running on the infrastructure or on a single machine. The results contain both the VISHNU identifier of the process and the DIET underlying middleware identifier.

OPTIONS

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

`-p` ***`machineId`*** The id of the 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:

"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]

EXAMPLE

To get the list of the vishnu processes that are running and monitored on machine_1:

vishnu_get_processes -p machine_1

11.2 vishnu_set_system_info

vishnu_set_system_info — updates the system information of a machine

Synopsis

`vishnu_set_system_info [-h] [-m memory] [-d diskSpace] machineId`

DESCRIPTION

This command with restricted access is used to set system information on a machine in the VISHNU database. The machine must first be registered using the UMS "addMachine" call. Using the machine identifier, information such as the total memory and available diskspace on the machine can be added.

OPTIONS

-h *help* help about the command.

-m *memory* Amount of RAM memory available on the machine (in Bytes).

-d *diskSpace* Amount of disk space available on the machine (in Bytes).

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]

"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]

EXAMPLE

To set the diskspace size to 300 on machine_1:

vishnu_set_system_info -d 300 machine_1

11.3 vishnu_set_system_threshold

vishnu_set_system_threshold — sets a threshold on a machine of a system

Synopsis

`vishnu_set_system_threshold` [`-h`] *`value machineId type handler`*

DESCRIPTION

This function allows an administrator to set a threshold. Each time an IMS server records the state of a machine, it checks the values defined, if a threshold is reached (over a use of the cpu or under the free memory or diskspace available), the administrator responsible for the threshold will receive an mail. These threshold will help the administrator to be aware of critical situations on a machine. Warning, a mail is sent for each time the threshold is reached, if a value swings around a threshold, the administrator may receive lots of emails depending on the update frequency.

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]

"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]

EXAMPLE

To set a threshold of type use of the CPU(value=1) of value 99% on machine_1 and handled by the user admin_1:

vishnu_set_system_threshold 99 machine_1 1 admin_1

11.4 vishnu_get_system_threshold

vishnu_get_system_threshold — gets a system threshold on a machine

Synopsis

`vishnu_get_system_threshold` [`-h`] [`-m` *machineId*] [`-t` *metricType*]

DESCRIPTION

This function allows an administrator to get the thresholds that may be defined on a machine. This may be used to check the parameters defined to monitor the machine. Each time a threshold is reached, a mail is sent. So checking the values of the threshold is important for the administrator to make sure they will not get tons of emails.

OPTIONS

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

`-m` ***`machineId`*** The id of the machine where the metric is defined.

`-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]

"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]

EXAMPLE

To get all the thresholds:

vishnu_get_system_threshold

11.5 vishnu_define_user_identifier

vishnu_define_user_identifier — defines the shape of the identifiers automatically generated for the users

Synopsis

`vishnu_define_user_identifier` [`-h`] *format*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the users. Once the format is defined, each time a user is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $NAME: The name of the user $UNAME: The name of the user $DAY: The day the user is added $MONTH: The month the user is added $YEAR: The year the user is added $SITE: The site the user is $TYPE: The 'U' symb to remind it is a user id

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]

"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]

EXAMPLE

To define the format to user_$CPT:

vishnu_define_user_identifier user_\$CPT

11.6 vishnu_define_machine_identifier

vishnu_define_machine_identifier — defines the shape of the identifiers automatically generated for the machines

Synopsis

`vishnu_define_machine_identifier` [-h] *format*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the machines. Once the format is defined, each time a machine is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $MANAME: The hostname of the machine $NAME: The hostname of the machine $DAY: The day the machine is added $MONTH: The month the machine is added $YEAR: The year the machine is added $SITE: The site the machine is $TYPE: The 'M' symb to remind it is a machine id

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]

"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]

EXAMPLE

To define the format to machine_$CPT:

vishnu_define_machine_identifier machine_\$CPT

11.7 vishnu_define_job_identifier

vishnu_define_job_identifier — defines the shape of the identifiers automatically generated for the jobs

Synopsis

`vishnu_define_job_identifier` [`-h`] *`format`*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the jobs submited through TMS. Once the format is defined, each time a job is submitted, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the job is submitted $MONTH: The month the job is submitted $YEAR: The year the job is submitted $TYPE: The 'J' symb to remind it is a job id

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]

"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]

EXAMPLE

To define the format to job_$CPT:

vishnu_define_job_identifier job_\$CPT

11.8 vishnu_define_transfer_identifier

vishnu_define_transfer_identifier — defines the shape of the identifiers automatically generated for the file transfers

Synopsis

`vishnu_define_transfer_identifier` [-h] *format*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the file transfers. Once the format is defined, each time a file transfer is done, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

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]

"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]

EXAMPLE

To define the format to transfer_$CPT:

vishnu_define_transfer_identifier transfer_\$CPT

11.9 vishnu_define_auth_identifier

vishnu_define_auth_identifier — defines the shape of the identifiers automatically generated for the authentication system

Synopsis

`vishnu_define_auth_identifier` [-h] *format*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the authentication. Once the format is defined, each time an authentication system is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

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]

"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]

EXAMPLE

To define the format to transfer_$CPT:

vishnu_define_auth_identifier LDAP_\$CPT

11.10 vishnu_load_shed

vishnu_load_shed — sheds load on a machine

Synopsis

`vishnu_load_shed` [`-h`] *`machineId loadShedType`*

DESCRIPTION

This function allows an administrator to shed load on a machine. Two modes are available: SOFT mode will cancel all the submitted jobs and file transfers for all VISHNU users (Note that jobs and file transfers not initiated through VISHNU will not be impacted). HARD mode will additionally stop all the VISHNU processes on the machine. If a user without administrator rights uses this function, all the user's jobs and file transfers will be cancelled on the machine. In the HARD mode, the stopped processes will not be automatically restarted. Type values: HARD = 1 SOFT = 2

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]

"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]

"If a component is unavailable" [301]

EXAMPLE

To make a hard load shedding on machine_1:

vishnu_load_shed machine_1 1

11.11 vishnu_set_update_frequency

vishnu_set_update_frequency — sets the update frequency of the IMS tables

Synopsis

`vishnu_set_update_frequency` [-h] *freq*

DESCRIPTION

This function allows an admin to set the update frequency. This frequency corresponds to how often the state of the machines is automatically polled by the IMS server. The value must be in seconds.

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]

"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]

EXAMPLE

To set the frequency to 100:

vishnu_set_update_frequency 100

11.12 vishnu_stop

vishnu_stop — To stop (and do not try to relaunch) a SeD

Synopsis

`vishnu_stop` [`-h`] *`processName machineId`*

DESCRIPTION

This function allows an admin to stop a VISHNU server on a machine. The stopped process will not be restarted automatically. The important parameters in the process are the names and the machine. The processName must be UMS, TMS, IMS or FMS , in upper case.

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]

"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]

EXAMPLE

To stop the UMS process on machine_1:

vishnu_stop UMS machine_1

11.13 vishnu_restart

vishnu_restart — To restart a SeD or a MA

Synopsis

`vishnu_restart [-h] [-v vishnuConf] [-t sedType] machineId`

DESCRIPTION

This function allows an admin to restart a VISHNU server on a machine. Warning when restarting a server, first it is tried to stop it, so if one is running it is stopped and then another is restarted.

OPTIONS

-h *help* help about the command.

-v *vishnuConf* The path to the vishnu configuration file.

-t *sedType* The type of the vishnu sed. The value must be an integer. Predefined values are: 0 (UNDEFINED), 1 (UMS), 2 (TMS), 3 (FMS), 4 (IMS).

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]

"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]

EXAMPLE

To restart using the configuration file ums.cfg an UMS sed on machine_1:

vishnu_restart -v /tmp/ums.cfg -t 1 machine_1

11.14 vishnu_define_work_identifier

vishnu_define_work_identifier — defines the shape of the identifiers automatically generated for the work

Synopsis

`vishnu_define_work_identifier` [`-h`] *`format`*

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the work. Once the format is defined, each time a work is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'W' symb to remind it is a file transfer id $NAME: The name of the work

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]

"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]

EXAMPLE

To define the format to W_$CPT:

vishnu_define_work_identifier W_\$CPT

Chapter 12

IMS C++ API Reference

12.1 getProcesses

getProcesses — displays the list of the VISHNU processes running on machines

Synopsis

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

DESCRIPTION

This command with restricted access is used to get the list of VISHNU server processes that are running on the infrastructure or on a single machine. The results contain both the VISHNU identifier of the process and the DIET underlying middleware identifier.

ARGUMENTS

sessionKey Input argument. The session key.

process Output argument. The list of the Vishnu processes on the machine.

options Input argument. The options to search for the processes.

EXCEPTIONS

The following exceptions may be thrown:

"If a parameter is invalid" [10]

12.2 setSystemInfo

setSystemInfo — updates the system information of a machine

Synopsis

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

DESCRIPTION

This command with restricted access is used to set system information on a machine in the VISHNU database. The machine must first be registered using the UMS "addMachine" call. Using the machine identifier, information such as the total memory and available diskspace on the machine can be added.

ARGUMENTS

sessionKey Input argument. The session key.

systemInfo Input argument. Contains system information to store in Vishnu database.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.3 setSystemThreshold

setSystemThreshold — sets a threshold on a machine of a system

Synopsis

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

DESCRIPTION

This function allows an administrator to set a threshold. Each time an IMS server records the state of a machine, it checks the values defined, if a threshold is reached (over a use of the cpu or under the free memory or diskspace available), the administrator responsible for the threshold will receive an mail. These threshold will help the administrator to be aware of critical situations on a machine. Warning, a mail is sent for each time the threshold is reached, if a value swings around a threshold, the administrator may receive lots of emails depending on the update frequency.

ARGUMENTS

sessionKey Input argument. The session key.

threshold Input argument. The threshold to set.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.4 getSystemThreshold

getSystemThreshold — gets a system threshold on a machine

Synopsis

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

DESCRIPTION

This function allows an administrator to get the thresholds that may be defined on a machine. This may be used to check the parameters defined to monitor the machine. Each time a threshold is reached, a mail is sent. So checking the values of the threshold is important for the administrator to make sure they will not get tons of emails.

ARGUMENTS

sessionKey Input argument. The session key.

value Output argument. The thresholds value.

options Input argument. The options for the threshold.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.5 defineUserIdentifier

defineUserIdentifier — defines the shape of the identifiers automatically generated for the users

Synopsis

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

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the users. Once the format is defined, each time a user is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $NAME: The name of the user $UNAME: The name of the user $DAY: The day the user is added $MONTH: The month the user is added $YEAR: The year the user is added $SITE: The site the user is $TYPE: The 'U' symb to remind it is a user id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.6 defineMachineIdentifier

defineMachineIdentifier — defines the shape of the identifiers automatically generated for the machines

Synopsis

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

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the machines. Once the format is defined, each time a machine is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $MANAME: The hostname of the machine $NAME: The hostname of the machine $DAY: The day the machine is added $MONTH: The month the machine is added $YEAR: The year the machine is added $SITE: The site the machine is $TYPE: The 'M' symb to remind it is a machine id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.7 defineJobIdentifier

defineJobIdentifier — defines the shape of the identifiers automatically generated for the jobs

Synopsis

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

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the jobs submited through TMS. Once the format is defined, each time a job is submitted, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the job is submitted $MONTH: The month the job is submitted $YEAR: The year the job is submitted $TYPE: The 'J' symb to remind it is a job id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.8 defineTransferIdentifier

defineTransferIdentifier — defines the shape of the identifiers automatically generated for the file transfers

Synopsis

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

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the file transfers. Once the format is defined, each time a file transfer is done, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.9 defineAuthIdentifier

defineAuthIdentifier — defines the shape of the identifiers automatically generated for the authentication system

Synopsis

int **vishnu::defineAuthIdentifier**(const string& sessionKey, const string& format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the authentication. Once the format is defined, each time an authentication system is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.10 loadShed

loadShed — sheds load on a machine

Synopsis

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

DESCRIPTION

This function allows an administrator to shed load on a machine. Two modes are available: SOFT mode will cancel all the submitted jobs and file transfers for all VISHNU users (Note that jobs and file transfers not initiated through VISHNU will not be impacted). HARD mode will additionally stop all the VISHNU processes on the machine. If a user without administrator rights uses this function, all the user's jobs and file transfers will be cancelled on the machine. In the HARD mode, the stopped processes will not be automatically restarted. Type values: HARD = 1 SOFT = 2

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine to stop.

loadShedType Input argument. Selects a load shedding mode (SOFT: stops all services and they can be restarted, HARD: stops all services, they cannot be restarted).

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

"If a component is unavailable" [301]

12.11 setUpdateFrequency

setUpdateFrequency — sets the update frequency of the IMS tables

Synopsis

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

DESCRIPTION

This function allows an admin to set the update frequency. This frequency corresponds to how often the state of the machines is automatically polled by the IMS server. The value must be in seconds.

ARGUMENTS

sessionKey Input argument. The session key.

freq Input argument. Frequency the data are updated, in second.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.12 stop

stop — To stop (and do not try to relaunch) a SeD

Synopsis

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

DESCRIPTION

This function allows an admin to stop a VISHNU server on a machine. The stopped process will not be restarted automatically. The important parameters in the process are the names and the machine. The processName must be UMS, TMS, IMS or FMS , in upper case.

ARGUMENTS

sessionKey Input argument. The session key.

process Input argument. The process to stop and do not try to restart anymore.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.13 restart

restart — To restart a SeD or a MA

Synopsis

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

DESCRIPTION

This function allows an admin to restart a VISHNU server on a machine. Warning when restarting a server, first it is tried to stop it, so if one is running it is stopped and then another is restarted.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine where to restart.

options Input argument. The option for the restart.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

12.14 defineWorkIdentifier

defineWorkIdentifier — defines the shape of the identifiers automatically generated for the work

Synopsis

int **vishnu::defineWorkIdentifier**(const string& sessionKey, const string& format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the work. Once the format is defined, each time a work is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'W' symb to remind it is a file transfer id $NAME: The name of the work

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

EXCEPTIONS

The following exceptions may be thrown:

"The database generated an error" [2]

"If a parameter is invalid" [10]

Chapter 13

IMS Python API Reference

13.1 VISHNU.getProcesses

VISHNU.getProcesses — displays the list of the VISHNU processes running on machines

Synopsis

ret, process=VISHNU.getProcesses(string sessionKey, ProcessOp options = ProcessOp());

DESCRIPTION

This command with restricted access is used to get the list of VISHNU server processes that are running on the infrastructure or on a single machine. The results contain both the VISHNU identifier of the process and the DIET underlying middleware identifier.

ARGUMENTS

sessionKey Input argument. The session key.

process Output argument. The list of the Vishnu processes on the machine.

options Input argument. The options to search for the processes.

RETURNED OBJECTS

errorCode (integer) Output parameter. Contains 0 on success and the error code on failure.

EXCEPTIONS

The following exceptions may be thrown:

UserException("If a parameter is invalid" [10])

13.2 VISHNU.setSystemInfo

VISHNU.setSystemInfo — updates the system information of a machine

Synopsis

ret=VISHNU.setSystemInfo(string sessionKey, SystemInfo systemInfo);

DESCRIPTION

This command with restricted access is used to set system information on a machine in the VISHNU database. The machine must first be registered using the UMS "addMachine" call. Using the machine identifier, information such as the total memory and available diskspace on the machine can be added.

ARGUMENTS

sessionKey Input argument. The session key.

systemInfo Input argument. Contains system information to store in Vishnu database.

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])

UserException("If a parameter is invalid" [10])

13.3 VISHNU.setSystemThreshold

VISHNU.setSystemThreshold — sets a threshold on a machine of a system

Synopsis

ret=VISHNU.setSystemThreshold(string sessionKey, Threshold threshold);

DESCRIPTION

This function allows an administrator to set a threshold. Each time an IMS server records the state of a machine, it checks the values defined, if a threshold is reached (over a use of the cpu or under the free memory or diskspace available), the administrator responsible for the threshold will receive an mail. These threshold will help the administrator to be aware of critical situations on a machine. Warning, a mail is sent for each time the threshold is reached, if a value swings around a threshold, the administrator may receive lots of emails depending on the update frequency.

ARGUMENTS

sessionKey Input argument. The session key.

threshold Input argument. The threshold to set.

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])

UserException("If a parameter is invalid" [10])

13.4 VISHNU.getSystemThreshold

VISHNU.getSystemThreshold — gets a system threshold on a machine

Synopsis

ret, value=VISHNU.getSystemThreshold(string sessionKey, ThresholdOp options);

DESCRIPTION

This function allows an administrator to get the thresholds that may be defined on a machine. This may be used to check the parameters defined to monitor the machine. Each time a threshold is reached, a mail is sent. So checking the values of the threshold is important for the administrator to make sure they will not get tons of emails.

ARGUMENTS

sessionKey Input argument. The session key.

value Output argument. The thresholds value.

options Input argument. The options for the threshold.

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])

UserException("If a parameter is invalid" [10])

13.5 VISHNU.defineUserIdentifier

VISHNU.defineUserIdentifier — defines the shape of the identifiers automatically generated for the users

Synopsis

ret=VISHNU.defineUserIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the users. Once the format is defined, each time a user is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $NAME: The name of the user $UNAME: The name of the user $DAY: The day the user is added $MONTH: The month the user is added $YEAR: The year the user is added $SITE: The site the user is $TYPE: The 'U' symb to remind it is a user id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])

13.6 VISHNU.defineMachineIdentifier

VISHNU.defineMachineIdentifier — defines the shape of the identifiers automatically generated for the machines

Synopsis

ret=VISHNU.defineMachineIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the machines. Once the format is defined, each time a machine is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $MANAME: The hostname of the machine $NAME: The hostname of the machine $DAY: The day the machine is added $MONTH: The month the machine is added $YEAR: The year the machine is added $SITE: The site the machine is $TYPE: The 'M' symb to remind it is a machine id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])

13.7 VISHNU.defineJobIdentifier

VISHNU.defineJobIdentifier — defines the shape of the identifiers automatically generated for the jobs

Synopsis

ret=VISHNU.defineJobIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the jobs submited through TMS. Once the format is defined, each time a job is submitted, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the job is submitted $MONTH: The month the job is submitted $YEAR: The year the job is submitted $TYPE: The 'J' symb to remind it is a job id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])

13.8 VISHNU.defineTransferIdentifier

VISHNU.defineTransferIdentifier — defines the shape of the identifiers automatically generated for the file transfers

Synopsis

ret=VISHNU.defineTransferIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the file transfers. Once the format is defined, each time a file transfer is done, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])

13.9 VISHNU.defineAuthIdentifier

VISHNU.defineAuthIdentifier — defines the shape of the identifiers automatically generated for the authentication system

Synopsis

ret=VISHNU.defineAuthIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the authentication. Once the format is defined, each time an authentication system is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'F' symb to remind it is a file transfer id

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])

13.10 VISHNU.loadShed

VISHNU.loadShed — sheds load on a machine

Synopsis

ret=VISHNU.loadShed(string sessionKey, string machineId, LoadShedType loadShedType);

DESCRIPTION

This function allows an administrator to shed load on a machine. Two modes are available: SOFT mode will cancel all the submitted jobs and file transfers for all VISHNU users (Note that jobs and file transfers not initiated through VISHNU will not be impacted). HARD mode will additionally stop all the VISHNU processes on the machine. If a user without administrator rights uses this function, all the user's jobs and file transfers will be cancelled on the machine. In the HARD mode, the stopped processes will not be automatically restarted. Type values: HARD = 1 SOFT = 2

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine to stop.

loadShedType Input argument. Selects a load shedding mode (SOFT: stops all services and they can be restarted, HARD: stops all services, they cannot be restarted).

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])

UserException("If a parameter is invalid" [10])

IMSVishnuException("If a component is unavailable" [301])

13.11 VISHNU.setUpdateFrequency

VISHNU.setUpdateFrequency — sets the update frequency of the IMS tables

Synopsis

ret=VISHNU.setUpdateFrequency(string sessionKey, int freq);

DESCRIPTION

This function allows an admin to set the update frequency. This frequency corresponds to how often the state of the machines is automatically polled by the IMS server. The value must be in seconds.

ARGUMENTS

sessionKey Input argument. The session key.

freq Input argument. Frequency the data are updated, in second.

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])

UserException("If a parameter is invalid" [10])

13.12 VISHNU.stop

VISHNU.stop — To stop (and do not try to relaunch) a SeD

Synopsis

ret=VISHNU.stop(string sessionKey, Process process);

DESCRIPTION

This function allows an admin to stop a VISHNU server on a machine. The stopped process will not be restarted automatically. The important parameters in the process are the names and the machine. The processName must be UMS, TMS, IMS or FMS , in upper case.

ARGUMENTS

sessionKey Input argument. The session key.

process Input argument. The process to stop and do not try to restart anymore.

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])

UserException("If a parameter is invalid" [10])

13.13 VISHNU.restart

VISHNU.restart — To restart a SeD or a MA

Synopsis

ret=VISHNU.restart(string sessionKey, string machineId, RestartOp options);

DESCRIPTION

This function allows an admin to restart a VISHNU server on a machine. Warning when restarting a server, first it is tried to stop it, so if one is running it is stopped and then another is restarted.

ARGUMENTS

sessionKey Input argument. The session key.

machineId Input argument. The id of the machine where to restart.

options Input argument. The option for the restart.

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])

UserException("If a parameter is invalid" [10])

13.14 VISHNU.defineWorkIdentifier

VISHNU.defineWorkIdentifier — defines the shape of the identifiers automatically generated for the work

Synopsis

ret=VISHNU.defineWorkIdentifier(string sessionKey, string format);

DESCRIPTION

This function allows an administrator to define the format of the identifier that will be automatically generated for the work. Once the format is defined, each time a work is added, the format will be used to define its identifier. The format can contain various variables, a variable is preceded by the '$' symbol. Moreover, the counter variable '$CPT' MUST be present in the format, otherwise it will be rejected. The available variables are: $CPT: a counter $DAY: The day the file transfer is done $MONTH: The month the file transfer is done $YEAR: The year the file transfer is done $TYPE: The 'W' symb to remind it is a file transfer id $NAME: The name of the work

ARGUMENTS

sessionKey Input argument. The session key.

format Input argument. The new format to use.

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])

UserException("If a parameter is invalid" [10])