VISHNU - Le guide de l'administrateur

COLLABORATORS

	TITLE : VISHNU - Le guide de l'administrateur	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Benjamin Isnard, Daouda Traoré, Eugène Pamba Capo-Chichi, Kevin Coulomb, and Ibrahima Cissé	November 21, 2012	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
1	08/03/2011	Version initiale pour le module UMS uniquement	K. COULOMB
2	18/03/2011	Ajout du lancement manuel avec forwarder et d'image de fichiers de configuration exemple	K. COULOMB
3	22/03/2011	Ajout des web services	K. COULOMB
4	11/05/2011	Réécriture du lancement avec fichier de configuration. Ajout d'un paragraphe pour le sendmail. Ajout de l'administration de TMS.	K. COULOMB, B.ISNARD
5	18/05/2011	Ajout du parametre de configuration dbConnectionsNb.	B.ISNARD
6	10/06/2011	Documentation pour IMS.	K.COULOMB
7	15/06/2011	Documentation pour FMS.	I.CISSE
8	22/06/2011	Ajout de l'option ENABLE_SWIG.	B.ISNARD
9	24/06/2011	Ajout de l'option vishnuMachineId dans les fichiers de configuration de UMS, IMS et FMS .	I.CISSE

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
10	13/07/2011	Mise à jour du document suites aux premiers retours .	K.COULOMB
11	11/08/2011	Prise en compte du gestionnaire de resources SLURM	D.TRAORE
12	23/08/2011	Ajout d'un lien sur des sites expliquant comment installer une base de données postgresql/mysql. Suppression des informations de mise à jour de la base (maintenant le script de création contient tout). Ajout d'une référence vers 'VISHNU_API'	K.COULOMB
13	14/12/2011	Mise à jour pour les nouveaux forwarder de DIET.	K.COULOMB
14	15/12/2011	Mise à jour de la section configuration des clés ssh requises pour FMS.	I.CISSE
15	16/12/2011	Ajout de la section configuration des clés ssh requises pour TMS.	D.TRAORE
16	30/01/2012	Modifie les requirements en fonction de la version de DIET	K. COULOMB
17	27/02/2012	Liste les libs de boost nécessaire	K. COULOMB
18	02/03/2012	Prise en compte du gestionnaire de resources LSF	D.TRAORE
19	22/03/2012	Ajout pour le support de LDAP	K. COULOMB
20	11/04/2012	Prise en compte du gestionnaire de resources Grid Engine	E. PAMBA CAPO-CHICHI
21	30/05/2012	Ajout des versions de LoadLeveler, GLIBC pour libcrypt et ssh	E. PAMBA CAPO-CHICHI
22	27/08/2012	Nouvelle compilation avec les batchs	K. COULOMB

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
23	01/10/2012	Ajout des prérequis en termes de connaissances systèmes et raffinement de la procédure d'installation	R. Chakode
24	21/11/2012	Première MAJ pour ZMQ	K. Coulomb

Contents

Chapter 1

Présentation du document

1.1 Objectifs du document

Ce document décrit l'installation, la configuration et l'administration des différents composants de la suite logicielle VISHNU.

1.2 Prérequis

Pour s'assurer un bon confort lors de la lecture de ce document, le lecteur doit au moins avoir des connaissances basiques en administration système en environnement GNU/Linux. En particulier et selon votre système d'exploitation, la maitrise d'un outil de gestion de paquets tels que *apt-get*, *dpkg*, *rpm*, *yum* ou *zypper* est vivement recommandée pour faciliter la recherche et l'installation des dépendances logicielles de VISHNU.

Par ailleurs, vous devez avoir bien compris l'architecture de déploiement de VISHNU. Cf. chapitre 4 du document [ARCH].

1.3 Structure du document

Ce document contient les parties suivantes:

- Définitions
- Installation
- Installation des web services
- Déploiement
- Administration
- Référence des commandes (en anglais)
- Référence de l'API C++ (en anglais)
- Référence de l'API Python (en anglais)

Chapter 2

Définitions

2.1 Acronymes

- DB : Database ou base de données, est un point central fournissant des mécanismes pour stocker et récupérer les données de manière efficace par les applicatons
- FMS : File Management Servic ou système de gestion de fichiers
- IMS : Information Management Service ou système de gestion d'information
- LDAP : Lightweight Directory Access Protocol, est un protocole de gestion d'annuaires
- SQL : Structured Query Language, est un langage avancée de requêtes sur les bases de données
- TMS : Task Management Service ou système de gestion de tâches
- UMS : User Management Service ou système de gestion des utilisateurs
- WS : Web Service ou service web
- ZMQ : Zero Message Queueing, une libraire de message

2.2 Références

- [ARCH] D1.1g-VISHNU Technical Architecture : description de l'architecture de l'application VISHNU
- [VISHNU_USERMANUAL] VISHNU User Manual : guide de l'utilisateur VISHNU.
- [VISHNU_API] VISHNU API : API VISHNU contenant les signatures et la définition des objets.

2.3 Glossaire

- Client FMS : Cela désigne les programmes permettant un accès à distance aux services du SeD FMS.
- Client IMS : Cela désigne les programmes permettant un accès à distance aux services du SeD IMS.
- Client TMS : Cela désigne les programmes permettant un accès à distance aux services du SeD TMS.
- Client UMS : Cela désigne les programmes permettant un accès à distance aux services du SeD UMS.
- Préfrontale : Cela désigne une machine mise avant la(les) frontale(s) des calculateurs.

- SeD fMS : Cela désigne le programme contenant et exécutant les services du module IMS.
- SeD IMS : Cela désigne le programme contenant et exécutant les services du module IMS.
- SeD TMS : Cela désigne le programme contenant et exécutant les services du module TMS.
- SeD UMS : Cela désigne le programme contenant et exécutant les services du module UMS.
- Sysfera-DS : Intergiciel open source développé par SysFera et composé d'un ensemble de logiciels

Chapter 3

Installation à partir des sources

3.1 Objectifs

Ce chapitre présentera comment installer VISHNU à partir des sources.

Nous décrirons comment installer et configurer la partie serveur et la partie client. L'installation de la partie client est indépendante de la partie serveur. Mais, à l'inverse, l'installation de la partie serveur nécessite l'installation de la partie client. Par ailleurs, il faut noter que le module UMS est requis pour faire fonctionner les modules FMS, IMS et TMS. Même si ces derniers peuvent être installés sur des machines indépendantes. Toutefois, pour qu'IMS puisse gérer le délestage, les différents modules doivent être installés sur la même machine.

3.2 Prérequis

Selon les modules sélectionnés, la compilation des sources de VISHNU nécessite les bibliothèques suivantes:

- Dépendances requises:
  - GCC : Version 4.4.3 ou ultérieure.
  - CMAKE : Version 2.8 ou ultérieure.
    GNU Make ou Ninja: Générateurs exécutables. Comme alternative à Make, le générateur Ninja est plus rapide et est utile pour réduire la durée de compilation.Surtout quand vous avez besoin de compiler plusieurs modules de VISHNU sur la même machine.
  - BOOST : Version 1.46.1 ou ultérieure. Au moins les modules *program_options*, *date_time, thread, filesystem, system, unit_test_framework, serialization, random* et *regex* doivent être installés
  - OMNIORB : Version 4.1.4 ou ultérieure.
  - GLIBC : Version 2.7 ou ultérieure.
  - openSSH : Version 4.2 ou ultérieure.
  - Bibliothèques de connexion à la base de données:
    VISHNU nécessite une base de données type MySQL ou PostGreSQL. Selon le type de base de données choisir, vous devez installer les API correspondantes en tenant compte des versions supportées:
    * API PostGreSQL (PGSQL-API) : Version 8.0 minimum.
    * API MySQL : Version 5.1 minimum.
- Dépendances optionelles pour le module UMS:
  - OpenLDAP : Version 2.4
- En fonction du Batch Scheduler sous-jacent à TMS, vous devez installer les APIs correspondantes :

  - Torque API : Version 2.3.6
  - IBM LoadLeveler API : Version 3.5.1.14
  - SLURM API : Versions 2.2.x, 2.3.x ou 2.4.x. Non testé avec les versions inférieures ou supérieures.
  - LSF API : Version 7.0.6.134609
  - Grid Engine API : Version 2011.11
- Dépendances optionnelles:
  - SWIG : Version 1.3.40 (ATTENTION : la version 2 de SWIG n'est pas supporté).
  - JAVA SDK : Version 1.6.
  - Python : Version 2.6
- Dépendances liées à IMS:
  - Sigar : Version 1.6.4, http://sourceforge.net/projects/sigar/files/sigar/1.6/
  - Supervisord : Version 3.0.a, pip install supervisord
  - XMLRPC : Version 2.1.19, apt-get install libxmlrpc

3.3 Compilation des sources

VISHNU utilise CMake comme système de construction et se conforme aux pratiques communément admises. Les options principales utilisables sont les suivantes:

- BUILD_TESTING, compile les tests si le flag est activé (OFF par défaut).
- CLIENT_ONLY, qui permet de ne compiler que les clients si le flag est activé (OFF par défaut). Sinon les clients et les serveurs sont compilés.
- CMAKE_INSTALL_PREFIX, répertoire d'installation (/usr/local par défaut sur les plateformes *nix)
- COMPILE_TMS, compile le module TMS si le flag est activé (OFF par défaut). Si ON, COMPILE_UMS doit aussi être à ON et les options VISHNU_BATCH et VISHNU_BATCH_VERSION doivent être remplies.
- COMPILE_IMS, compile le module IMS si le flag est activé (OFF par défaut). Si ON, COMPILE_UMS, COMPILE_TMS et COMPILE_FMS doivent aussi être à ON.
- COMPILE_FMS, compile le module FMS si le flag est activé (OFF par défaut). Si ON, COMPILE_UMS doit aussi être à ON.
- COMPILE_UMS, compile le module UMS si le flag est activé (ON par défaut).
- ENABLE_PYTHON, qui permet d'activer la compilation du code PYTHON (OFF par défaut).
- ENABLE_JAVA, qui permet d'activer la compilation des sources JAVA pour les web services (OFF par défaut).
- VISHNU_USE_LDAP, qui permet d'activer le support LDAP pour l'authentification (OFF par défaut).
- ENABLE_SWIG, qui permet d'activer la generation du code des adapteurs Python et Java (OFF par défaut). Cette option doit obligatoirement être activée si on choisit de ne pas compiler tous les modules VISHNU c'est-à-dire si au moins l'une des options COMPILE_UMS, COMPILE_TMS, COMPILE_FMS ou COMPILE_IMS est à OFF. Si l'option ENABLE_SWIG est activée, le package SWIG (voir paragraphe 'Prérequis') est nécessaire pour compiler.
- VISHNU_BATCH, compile le SeD TMS avec le batch correspondant (TORQUE par défaut).

  Les variables (BATCHNAME)_INCLUDE_DIR et (BATCHNAME)_LIB doivent être renseignées.
  - LOADLEVELER_INCLUDE_DIR indique le chemin absolu vers le fichier llapi.h
  - LOADLEVELER_LIB indique le chemin absolu jusqu'à la librairie llapi.so
  - TORQUE_INCLUDE_DIR indique le chemin absolu vers le fichier pbs_ifl.h

  - TORQUE_LIB indique le chemin absolu jusqu'à la librairie libtorque.so
  - SLURM_INCLUDE_DIR indique le chemin absolu vers le fichier slurm/slurm.h
  - SLURM_LIB indique le chemin absolu jusqu'à la librairie libslurm.so
  - LSF_INCLUDE_DIR indique le chemin absolu vers le fichier lsf/lsbatch.h
  - LSBATCH_LIB indique le chemin absolu jusqu'à la librairie libbat.so
  - LSF_LIB indique le chemin absolu jusqu'à la librairie liblsf.so
  - SGE_ROOT indique le chemin absolu jusqu'au répertoire d'installation de Grid Engine
  - SGE_INCLUDE_DIR indique le chemin absolu vers le fichier drmaa.h
  - SGE_BIN_DIR indique le chemin absolu jusqu'aux binaires de Grid Engine
  - SGE_LIB indique le chemin absolu jusqu'à la librairie libdrmaa.so
- VISHNU_BATCH_VERSION indique la version du batch scheduler utilisée

Par exemple, les étapes suivantes permettent d'installer les clients et les serveurs d'UMS et TMS dans /opt/vishnu. TORQUE est utilisé comme backend à TMS. L'API Python est également activée.

- 1. Créer un répertoire build à la racine du projet et se placer dedans

  $ *mkdir build*

  $ *cd build*
- 2. Générer le Makefile
  - En utilisant 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
  - En utilisant Ninja : changement du générateur via le 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. Lancer la compilation
  - En utilisant Make

    $ *make -j 2*

    L'option -j 2 permet de lancer la compilation avec deux processus.
  - En utilisant Ninja

    $ *ninja -j 2*

    L'option -j 2 permet de lancer la compilation avec deux processus.
- 4. Installer les binaires

  $ *make install*

  Cette étape peut nécessiter des droits d'administrateur.

Note: pensez à ajouter le répertoire d'installation dans le $PATH.

Chapter 4

Configuration de la base de données

Les fichiers de configuration de la DB sont disponibles dans le répertoire core/database du package d'installation VISHNU. Seules des bases de données PostgreSQL ou MySQL sont actuellement supportées. Une seule base de données est nécessaire et suffisante pour l'ensemble des composants d'une infrastructure VISHNU (UMS, TMS, IMS et IMS).

4.1 Utiliser une base de données MySQL

Nous supposons ici que vous avez déjà une installation de MySQL opérationnelle. Si ce n'est pas le cas, référer à la documentation officielle pour procéder à l'installation.

Vous devez également disposer des droits d'administration sur cette installation (connexion à MySQL en tant l'utilisateur 'root').

Vous aurez besoin des scripts *mysql_create.sql* et *database_init.sql* disponibles dans le dossier *./core/database* de l'arborescence des sources. Le premier script (*mysql_create.sql*) sert créer les tables de la base de données tandis que le second sert à initialiser les données.

Pour la création de la base de données VISHNU, suivez les étapes suivantes :

- Se connecter au serveur MySQL en tant que root :

  $ *mysql -h mysql@server- -u root -p* # replacer mysql@server par l'adresse de votre serveur MySQL
- Créer la base de données Vishnu

  $ *create database vishnu ;*
- Se connecter sur la base de données

  $ *use vishnu ;*
- Créer les tables

  $ *source /path/to/mysql_create.sql*
- Initialiser la base de données

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

4.2 Utiliser une base de données PostGreSQL

Nous supposons également que vous avez déjà une installation de PostGreSQL opérationnelle. Si ce n'est pas le cas, référer à la documentation officielle pour procéder à l'installation.

Vous devez également disposer des droits d'administration sur cette installation (accès en tant qu'utilisateur système 'postgres').

Pour une nouvelle installation de la base de données, les scripts *postgre_create.sql* et *database_init.sql* doivent respectivement être utilisés pour créer et initialiser la base de données.

Pour créer et initialiser la base de données, exécuter les étapes suivantes à partir de votre serveur PostGreSQL :

- Se connecter en tant que l'utilisateur 'postgres' (root):
  $ *su postgres*
- Créer la base de données vishnu
  $ *createdb vishnu ;*
- Se connecter à la base de données
  $ *psql vishnu ;*
- Créer le schéma de la base de données
  $ \i */path/to/mysql_create.sql*
- Initialiser la base de données
  $ \i */path/to/database_init.sql*

4.3 Utilisation de LDAP

Nous supposons que vous avez déjà une installation de LDAP opérationnelle. Si ce n'est pas le cas, référer à la documentation officielle pour procéder à l'installation. Vous pouvez également installer ces dépendances à partir du gestionnaire de paquets de votre système. Sur les systèmes basés sur Debian par exemple, vous devez installer les paquets suivants: *slapd, libldap-2.4-2, libldap2-dev* et *ldap-utils*

Pour pouvoir utiliser l'authentification avec LDAP, le flag de compilation VISHNU_USE_LDAP doit être activé à la compilation. Une fois VISHNU compilé, une option de configuration du serveur UMS doit être mise dans le fichier de configuration. Cette option est nommée 'authenticationType' et peut actuellement supporter 4 valeurs :

- UMS : Authentifie uniquement en utilisant la base de données
- LDAPUMS : Pour chaque couple (nom d'utilisateur, mot de passe), essaye d'authentifier avec LDAP en premier puis avec UMS
- UMSLDAP : Pour chaque couple (nom d'utilisateur, mot de passe), essaye d'authentifier avec UMS en premier puis avec LDAP
- LDAP : Authentifie uniquement en utilisant LDAP

Chapter 5

Installation des web services

5.1 Pré-requis

- Installer **Java 1.6** (commande 'sudo apt-get install openjdk-6-jdk' ou bien 'sudo apt-get install sun-java6-jdk' sur Linux Debian) et vérifier que la variable JAVA_HOME est bien définie et contient le répertoire racine de l'installation de java.
- Installer les modules désirés avec les options -DENABLE_JAVA=ON et -DENABLE_SWIG (**Note** : Ce pré-requis est optionnel car les jars sont déjà fournis dans la distribution des web services).
- Installer **Maven 2** pour compiler les jars (**Note** : Ce pré-requis est optionnel car les jars sont déjà fournis dans la distribution des web services. Par ailleurs la compilation avec maven nécessite une connexion internet).

5.2 Installation de JBoss

- Télécharger le package JBOSSAS : (la version binaire est disponible) sur http://www.jboss.org/jbossas/downloads => version 5.1.0.GA
- Télécharger le package JBOSSWS : (la version binaire est disponible) sur http://www.jboss.org/jbossws/downloads/ => version 3.3.1.GA
- Décompresser l'archive du package JBOSSAS 5.1.0
- Définir la variable d'environement JBOSS_HOME sur le répertoire décompressé. Par exemple dans le .bashrc : 'export JBOSS_HOME=/home/toto/jboss-5.1.0.GA'
- Décompresser l'archive du package JBOSSWS 3.3.1 et aller dans le répertoire créé
- Copier le fichier 'ant.properties.example' en 'ant.properties'
- Editer le fichier nouvellement créé 'ant.properties'. Mettre la valeur de la variable jboss510.home à la valeur du répertoire de JBOSSAS (même valeur que dans la variable JBOSS_HOME). Il faut noter que si une autre version que la 5.1.0 de jboss a été prise, il faut modifier la variable correspondant à cette version
- Lancer la commande 'ant deploy-jboss510'. Si une autre version de jboss a été prise, il faut faire la commande de la version correspondante
- Le lancement du serveur se fait en lançant le script JBOSS_HOME/bin/run.sh. Pour que le serveur jboss soit accessible depuis une autre machine, utiliser l'option '-b adresseIP' où adresseIP représente l'adresse IP du serveur, et vérifier que le firewall du serveur autorise l'accès au port 8080.
- **Vérification de l'installation:** Pour vérifier que le serveur JBoss est bien démarré et que le module web services est activé, lancer un navigateur internet sur le serveur et se connecter sur l'adresse 'http://localhost:8080/jbossws' (ou 'http://adresseIP:8080/jboss et vérifier que la page affiche bien la version du module web services (jbossws-cxf-3.3.1.GA).

5.3 Installation des modules WS dans JBoss

5.3.1 Fichiers à installer

- VISHNULib-1.0-SNAPSHOT.jar
  - Contient les classes internes faisant le lien JAVA(JNI)/C++.
  - A copier dans **JBOSS_HOME/server/default/lib**.
  - Le changer implique un redémarrage du serveur JBOSS.
  - Compilation (si nécessaire)
    - Aller dans le répertoire VISHNULib
    - Faire 'mvn install' (peut être long la première fois)
    - Le fichier .jar produit se trouve dans le répertoire target/
- WSAPI.jar
  - Contient les classes issues du WSDL et l'implémentation des WS.
  - A mettre dans **JBOSS_HOME/server/default/deploy**.
  - Peut être mis à jour sans redémarrer le serveur JBOSS.
  - Compilation (si nécessaire)
    - Aller dans le répertoire WSAPI, avoir le jar VISHNULib-1.0-SNAPSHOT.jar de déjà fait
    - Faire 'mvn install' (peut être long la première fois)
    - Le fichier .jar produit se trouve dans le répertoire target
    - Renommer le jar en WSAPI.jar avant de le mettre dans jboss, ceci est nécessaire
- libVISHNU.so
  - Nécessaire pour le fonctionnement des WS, cette librairie dynamique est obtenue en compilant les modules VISHNU désirés avec l'option ENABLE_JAVA.
  - Le fichier est installé par défaut par le package VISHNU UMS dans '/usr/local/lib' et il n'est pas nécessaire de le copier.
  - S'il y a un problème de déploiement dans le serveur jboss, vérifier qu'elle est bien accessible et dans le LD_LIBRARY_PATH.

5.3.2 Variables d'environnement à définir

- VISHNU_CONFIG_FILE : contient le chemin complet du fichier de configuration client de VISHNU. Se référer au guide d'installation du client [VISHNU_USER_GUIDE] pour connaître le contenu de ce fichier. Si l'exécution échoue avec un message d'erreur lié à initialisation de la librairie lors du connect, vérifier le contenu de cette variable.
- LD_LIBRARY_PATH : contient les chemins des répertoires contenant les librairies VISHNU, notamment libVISHNU.so.

5.3.3 Lancement du serveur JBoss avec le module WS

- Après installation des fichiers définis au paragraphe précédent, le serveur JBoss doit être redémarré en lançant le script JBOSS_HOME/bin/run.sh (avec les options indiquées au paragraphe sur le serveur JBoss).
- **Vérification de l'installation:** Pour vérifier que le serveur JBoss est bien démarré et que le module UMS WS est activé, lancer un navigateur internet sur le serveur et se connecter sur l'adresse 'http://localhost:8080/jbossws/services' (ou 'http://adresseIP:8080/jbossws/services') et vérifier que le "service endPoint" : **VISHNUUMSPortImpl** est actif.

Chapter 6

Déploiement de VISHNU

6.1 Les composants logiciels

•

•

6.2 Lancement sur un même réseau

Les modules peuvent être lancés manuellement. Le déploiement se fera en suivant la configuration montrée sur la figure 4.1 du document [ARCH]. Pour simplier les choses, on supposera que les divers éléments dans le carré "dedicated to VISHNU" vont être déployés sur la même machine que l'on va nommer **préfrontale**.

NOTES IMPORTANTES:

- Il est préférable d'éviter de lancer le même SeD sur la même machine.
- Il existe un bug connu sur debian (entre autre) avec boost file system, utilisé par VISHNU. Le rapport de bug est ici et le bug est actuellement ouvert: https://svn.boost.org/trac/boost/ticket/4688. Si lors du lancement d'un SeD, le message d'erreur suivant apparait : std::runtime_error: locale::facet::_S_create_c_locale name not valid, faire un "export LANG=C" et cela devrait régler le problème.
- Les fichiers de configuration de tout les éléments mentionnés ici sont présentés dans une section plus loin exemple avec les détails de chacun

Une fois ces mises en gardes précisées, passons au lancement de l'environnement. Pour démarrer avec succès votre environnement VISHNU, suivez rigoureusement les étapes suivantes:

1. Vérifier que la base de données (PostGreSQL ou MySQL) a été bien configurée et initialisée comme décrit précédemment. De plus, vérifier vous pouvez vous connecter à la base de données depuis les différents serveurs où sont installés les modules de VISHNU.
2. Sur la machine hôte de Torque, lancer le serveur (pbs_serv), le scheduler (pbs_sched) et l'ordonnanceur (pbs_mom) de Torque.
3. Sur la machine hôte de SLURM, lancer les serveurs *slurmd, slurmctld* et *slurmdbd*.
4. Sur la machine hôte de LSF, lancer les exécutables hostsetup *lsfstartup*.
5. Sur la machine hôte de Grid Engine, lancer les exécutables *sge_qmaster* et *sge_execd*.

6. Optionnel: Sur une machine lancer le dispatcher, le fichier de configuration 'config' doit être à la fin et est optionnel.

   dispatcher tcp://127.0.0.1:5560 tcp://127.0.0.1:5561 -c config

   Dans cette commande, on demande au dispatcher de démarrer en recevant les requêtes clients sur l'URI tcp://127.0.0.1:5560, de recevoir les requêtes de contrôle sur l'URI tcp://127.0.0.1:5561, et de démarrer en utilisant les serveurs définis dans le fichier config.

7. Sur chaque machine serveur, lancer supervisord avec son fichier de configuration 'config' correspondant. Le fichier de configuration de supervisord contient le démarrage de chaque serveur vishnu sur la machine en question.

   supervisord -c config

8. Les modules de VISHNU sont prêts à être utilisés. Pour ce faire, un client doit se connecter et soumettre des requêtes à VISHNU au moyen des commandes UMS, TMS, FMS et IMS (voir le manuel de l'utilisateur VISHNU pour plus d'informations sur les commandes utilisateurs disponibles).

6.3 Présenter le cas multi-réseaux

6.4 Exemple de fichier du dispatcher

Le fichier se compose de 3 colonnes, la première le serveur, la seconde son URI, la troisième l'identifiant de machine VISHNU. Dans l'exemple ci-dessous on a les 4 serveurs sur une URI pour une machine

```
TMS tcp://127.0.0.1:5557 machine_1
UMS tcp://127.0.0.1:5555 machine_1
FMS tcp://127.0.0.1:5556 machine_1
IMS tcp://127.0.0.1:5558 machine_1
```

6.5 Exemple de fichier du dispatcher

```
[inet_http_server]         ; inet (TCP) server disabled by default
port=127.0.0.1:9001        ; (ip_address:port specifier, *:port for all if
[supervisord]
logfile=/tmp/supervisord.log ; (main log file;default $CWD/supervisord.log)
logfile_maxbytes=50MB        ; (max main logfile bytes b4 rotation;default 50MB)
logfile_backups=10           ; (num of main logfile rotation backups;default 10)
loglevel=info                ; (log level;default info; others: debug,warn,trace)
pidfile=/tmp/supervisord.pid ; (supervisord pidfile;default supervisord.pid)
nodaemon=false               ; (start in foreground if true;default false)
minfds=1024                  ; (min. avail startup file descriptors;default 1024)
minprocs=200                 ; (min. avail process descriptors;default 200)
[rpcinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:make_main_rpcinterface

[supervisorctl]
serverurl=http://127.0.0.1:9001 ; use an http:// url to specify an inet socket,  ↩
    same as inet_http_server

; Attention bien respecter les nommages 'umssed', 'imssed', 'tmssed', 'fmssed'  ↩
    sinon ims ne pourra pas les gérer
[program:umssed] ; Définit un process à surveiller, son alias est umssed
command=umssed ~/conf/ums_config.cfg ; Commande comme on ferait dans un terminal
[program:fmssed] ; Définit un process à surveiller, son alias est fmssed
command=fmssed ~/conf/fms_config.cfg
[program:imssed] ; Définit un process à surveiller, son alias est imssed
command=imssed ~/conf/ims_config.cfg
[program:tmssed] ; Définit un process à surveiller, son alias est tmssed
command=tmssed ~/conf/tms_config.cfg
```

6.6 Exemple de fichier de configuration d'un SeD UMS

```
# Configuration of the VISHNU UMS SeD
vishnuId=1
databaseType=postgresql # postgresql ou mysql selon la DB
databaseHost=localhost # Hote pour la DB
databaseName=vishnu # Nom de la base ou on se connecte
databaseUserName=vishnu_user # Nom d'utilisateur pour se connecter à la DB
databaseUserPassword=vishnu_user # Mot de passe pour se connecter à la DB
databaseConnectionsNb=5 # Nombre de connexions parallèles à la DB
sendmailScriptPath=/usr/local/sbin/sendmail.py # Script installé par VISHNU dans ↩
    install_dir/sbin
vishnuMachineId=machine_1 # Optionnel pour UMS, le machine id de VISHNU ou le serveur ↩
    tourne
      authenticationType=LDAP # Le type d'authentification utilisé
      uri=tcp://127.0.0.1:5555 # L'URI ou le serveur est accessible par les clients
      uriDispatcherSub=tcp://127.0.0.1:5561 # Adresse d'inscription
      urlSupervisor=http://127.0.0.1:9001 # URL ou se trouve le supervisor du serveur en ↩
          question sur la machine
```

6.7 Exemple de fichier de configuration d'un SeD TMS

```
# Configuration of the VISHNU TMS SeD
vishnuId=1
databaseType=postgresql # postgresql ou mysql selon la DB
databaseHost=localhost # Hote pour la DB
databaseName=vishnu # Nom de la base ou on se connecte
databaseUserName=vishnu_user # Nom d'utilisateur pour se connecter à la DB
databaseUserPassword=vishnu_user # Mot de passe pour se connecter à la DB
databaseConnectionsNb=5 # Nombre de connexions parallèles à la DB
sendmailScriptPath=/usr/local/sbin/sendmail.py # Script installé par VISHNU dans ↩
    install_dir/sbin
vishnuMachineId=machine_1 # Le machine id de VISHNU ou le serveur tourne
      uri=tcp://127.0.0.1:5557 # L'URI ou le serveur est accessible par les clients
      uriDispatcherSub=tcp://127.0.0.1:5561 # Adresse d'inscription
      urlSupervisor=http://127.0.0.1:9001 # URL ou se trouve le supervisor du serveur en ↩
          question sur la machine
# Configuration propres à TMS
# batchSchedulerType=SLURM si le batchScheduler est de type SLURM
# batchSchedulerType=LSF si le batchScheduler est de type LSF
# batchSchedulerType=SGE si le batchScheduler est de type Grid Engine
batchSchedulerType=TORQUE
intervalMonitor = 1 # Intervalle de temps en seconde pour mettre à jour les infos du ↩
    batch scheduler
```

6.8 Exemple de fichier de configuration d'un SeD FMS

```
# Configuration of the VISHNU FMS SeD
vishnuId=1
databaseType=postgresql # postgresql ou mysql selon la DB
databaseHost=localhost # Hote pour la DB
databaseName=vishnu # Nom de la base ou on se connecte
databaseUserName=vishnu_user # Nom d'utilisateur pour se connecter à la DB
databaseUserPassword=vishnu_user # Mot de passe pour se connecter à la DB
databaseConnectionsNb=5 # Nombre de connexions parallèles à la DB
```

```
sendmailScriptPath=/usr/local/sbin/sendmail.py # Script installé par VISHNU dans ↩
    install_dir/sbin
vishnuMachineId=machine_1 # Le machine id de VISHNU ou le serveur tourne
      uri=tcp://127.0.0.1:5556 # L'URI ou le serveur est accessible par les clients
      uriDispatcherSub=tcp://127.0.0.1:5561 # Adresse d'inscription
      urlSupervisor=http://127.0.0.1:9001 # URL ou se trouve le supervisor du serveur en ↩
          question sur la machine
# Configuration propres à FMS
intervalMonitor = 1 # Intervalle de temps pour regarder l'état des transferts de fichiers ↩
     en cours
```

6.9 Configuration de l'envoi des emails par VISHNU

Le processus UMS SeD utilise le fichier 'sendmail.py' (fourni dans l'installation VISHNU, dans le sous-répertoire sbin/) pour envoyer des emails aux utilisateurs lors de certaines opérations. Ce fichier peut être modifié par l'administrateur afin de s'adapter à la méthode d'envoi d'email propre au serveur sur lequel est installé le SeD. Par défaut, la configuration fournie se connecte sur le serveur SMTP de 'localhost' sur le port 25, sans authentification.

Les paramètres suivants peuvent être configurés dans le script sendmail.py :

Option	**Ligne du script sendmail.py à modifier**
login	`parser.add_option("--login", dest="login", help="", default="[ ↩ login_utilisateur]")`
password	`parser.add_option("--password", dest="password", help="smtp password", ↩ default="[password_utilisateur]")`
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)`

6.10 Configuration des clés privées/publiques ssh requises pour FMS

Toutes les commandes éxécutées par le SeD FMS sont lancées via ssh sous le nom de l'utilisateur ayant émis la requête. Les services FMS sont de deux types : il y a ceux qui n'impliquent qu'une machine distante : Exemple getFilesInfo,listDir,etc.. et ceux qui impliquent au moins deux machines distantes : machine source et destination pour les transferts de fichiers.

- Dans le premier cas le SeD se connecte sur la machine distante et effectue la commande. Par conséquent la clé publique du SeD doit être ajoutée au fichier authorized_keys ($HOME/.ssh/authorized_keys) de l'utilisateur de la machine distante concernée.
- Dans le second cas, deux connexions ssh sont nécessaires. Le SeD se connecte sur la machine source et lance le transfert (seconde connexion) vers la machine destination. Par conséquent:

- la clé publique du SeD doit être ajoutée au fichier authorized_key de la machine source pour permettre la première connexion.
- La machine source doit pouvoir se connecter sur la machine destination par ssh, avec la clé privée enregistrée dans la base de VISHNU lors de l'ajout du compte (local account) liant la machine source à VISHNU. Par ailleurs si le mécanisme d'agent forwarding (de ssh) est activée entre ces différentes machines, il n'est alors plus nécessaire qu'il y ait un autre couple de clés entre la machine source et destination.

En somme, il est alors obligatoire que la clé publique du SeD soit ajoutée à tous les comptes utilisateurs des machines impliquées par les requêtes FMS. Toutes les clés protégées par des passphrases devront être stockées par un agent ssh pour permettre les authentifications automatiques.

6.11 Configuration des clés privées/publiques ssh requises pour TMS

Les commandes de soumission, d'annulation et de récupération des résultats de jobs éxécutées par le SeD TMS sont lancées via ssh sous le nom de l'utilisateur ayant émis la requête. Pour pouvoir exécuter ces commandes correctement, la clé publique du compte dédié au SeD TMS doit être ajoutée au fichier authorized_keys ($HOME/.ssh/authorized_keys) de l'utilisateur. Toutes les clés protégées par des passphrases devront être stockées par un agent ssh pour permettre les authentifications automatiques.

6.12 Test d'exécution d'un service depuis une machine client par shell

1. Une fois que la plateforme a été installée, se mettre sur un poste client avec VISHNU d'installé. Se référer au document [VISHNU_USERMANUAL] pour l'installation de la partie client.
2. Exporter la variable d'environnement VISHNU_CONFIG_FILE dans un script de configuration client. Se référer au guide d'installation du client [VISHNU_USER_GUIDE] pour connaître le contenu d'un fichier client.
3. Ouvrir une session VISHNU

   $ *vishnu_connect -u user*

   Remplacer 'user' par un vrai identifiant utilisateur. Par défaut, VISHNU est installé avec un utilisateur 'root' ayant tous les droits sur la plateforme (ID: 'root', Mot de passe: 'vishnu_user').
4. Entrer le mot de passe puis valider
5. Sur le client, un affichage doit signaler que le service a réussi. Dans le terminal ou le SeD UMS a été lancé et dans le terminal ou le MA tourne, selon le niveau de verbosité, plus ou moins d'informations, concernant le service effectué, doivent apparaitre. Le message affiché contient au moins une ligne similaire : **sessionId: root-2011-Jul-11-14:22:14.403491:86690**, qui indique l'identifiant de la session ouverte.
6. Fermer la session.

   $ *vishnu_close*

   Aucune erreur ne doit être remontée.

Chapter 7

Administration

7.1 Présentation

Le module UMS correspond à la gestion des utilisateurs et des machines de VISHNU. Il permet aussi de sauvegarder la configuration de VISHNU à chaud et de la restaurer si besoin est. Dans toute la suite du chapitre, on supposera que l'utilisateur est déjà connecté avec un compte administrateur de VISHNU pour pouvoir réaliser ces manipulations. De plus, on présentera l'utilisation des commandes depuis le shell, mais cela reste valable depuis les API Python ou C++.

L'API est disponible dans le document [VISHNU_API]

7.2 Gestion des utilisateurs (UMS)

1. L'ajout d'un utilisateur se fait à l'aide de la commande 'vishnu_add_user'. Elle prend en paramètre le prénom de l'utilisateur, son nom de famille, les droits qui lui sont associés dans VISHNU (administrateur ou simple utilisateur) et son adresse de couriel. Tout ces paramètres sont obligatoires. Un privilège à 1 signifie administrateur, un privilège à 0 signifie un utilisateur. L'identifiant de l'utilisateur est généré et renvoyé.

2. La mise à jour d'un utilisateur ne peut être faite que par un administrateur. Cette mise à jour se fait avec un appel à la commande 'vishnu_update_user' et permet de modifier les paramètres de l'ajout (nom, prénom, statut, couriel). Il faut avoir l'identifiant de l'utilisateur (généré lors de la création de l'utilisateur) pour le désigner lors de la mise à jour.

   Note : le changement du statut d'un utilisateur à l'état "INACTIVE" correspond à un blocage de son compte.

3. La suppression d'un utilisateur efface toutes les informations liées à l'utilisateur de la base de donnée. Cette suppression se fait à l'aide de la commande 'vishnu_delete_user'.

4. La liste des utilisateurs ne peut être faite que par un administrateur. Cela se fait avec la commande 'vishnu_list_user'. Cette commande peut prendre en paramètre l'identifiant d'un utilisateur pour n'avoir les informations que concernant cet utilisateur.

5. Seul un administrateur peut réinitialiser le mot de passe d'un utilisateur de VISHNU. Pour ce faire, il doit appeller la commande 'vishnu_reset_password' en fournissant l'identifiant de l'utilisateur dont l'administrateur veut réinitialiser le mot de passe. Le nouveau mot de passe est temporaire et renvoyé par la commande. Lors de la prochaine connexion, l'utilisateur devra changer son mot de passe avec 'vishnu_change_password'.

7.3 Gestion des machines (UMS+IMS)

1. L'ajout d'une machine se fait à l'aide de la commande 'vishnu_add_machine'. Cette commande prend en paramètre le nom de la machine, le site ou elle se trouve, le language de la description qui sera donnée pour la machine, le fichier contenant la clé publique et la description. Ces paramètres sont obligatoires, en passant par le shell, la description n'a pas besoin

d'être fournie en paramètre mais elle est alors demandée à l'administrateur avant d'ajouter la machine. A la fin de l'ajout, l'identifiant généré pour la machine est renvoyé.

2. La mise à jour d'une machine se fait à l'aide de la commande 'vishnu_update_machine' et permet de modifier les paramètres mis lors de l'ajout de la machine. Il faut utiliser l'identifiant de la machine pour l'identifier lors de la mise à jour.

   Note : le changement du statut d'une machine à l'état "INACTIVE" correspond à un blocage de la machine. Cela rend la machine inaccessible aux utilisateurs de VISHNU mais toujours visible pour les administrateurs.

3. La suppression d'une machine se fait à l'aide de la commande 'vishnu_delete_machine' avec l'identifiant de la machine à supprimer. Cela supprime la machine de la base de données, ainsi que toutes les informations qui y sont attachées (Attention : cette commande est irréversible).

4. Les utilisateurs peuvent lister les machines, mais un administrateur a en plus une option qui est l'identifiant d'un utilisateur. Ceci lui permet de lister les machines sur lesquelles l'utilisateur a un compte VISHNU.

5. La mise à jour d'informations système d'une machine se fait à l'aide de la commande 'vishnu_set_system_info' et permet d'ajouter ou modifier des informations système d'une machine. Il faut utiliser l'identifiant de la machine pour l'identifier lors de la mise à jour.

7.4 Gestion de la plateforme (UMS)

1. L'administrateur peut faire une sauvegarde à chaud à un moment donné de VISHNU. Ceci sauvegarde les utilisateurs, les machines et les comptes des utilisateurs. Le fichier, dans lequel la configuration est, est retourné par la fonction. La fonction est 'vishnu_save_configuration', pas besoin de paramètres.

2. L'administrateur peut recharger une configuration précédente de VISHNU à l'aide de la commande 'vishnu_restore_configuration' qui a besoin du fichier de sauvegarde pour recharger la base. Avant de pouvoir lancer cette restauration, tous les utilisateurs de VISHNU doivent être déconnectés.

3. Un administrateur peut également définir les valeurs par défaut des options de VISHNU pour tout les utilisateurs (ces options sont le temps de déconnexion par défaut et le type de fermeture d'une session par défaut). Cela se fait en appellant 'vishnu_configure_default_option' en donnant le nom de l'option et sa nouvelle valeur.

4. Un administrateur peut ajouter ou modifier un système d'authentification. Par exemple il peut ajouter différents LDAP pour authentifier ses utilisateurs. Actuellement, pour un même LDAP, si les utilisateurs sont dans des branches différentes il faut ajouter un système d'authentification par branche. Ceci afin de faciliter la connexion de l'utilisateur, pour qu'il ne donne aucune information concernant son arbre LDAP, juste son login. C'est l'administrateur lorsqu'il crée son système d'authentification qui doit remplir le champs ldapbase avec le chemin complet dans l'arbre LDAP à utiliser pour authentifier un utilisateur. Lorsqu'un administrateur rempli le chemin menant aux utilisateurs, il doit remplacer le nom de l'utilisateur par la chaîne de caractère \$USERNAME, cette chaîne sera remplacée par vishnu par le login de l'utilisateur que l'on cherche à authentifier. Fonctions associées : vishnu_add_auth_system, vishnu_update_auth_system, vishnu_delete_auth_system

7.5 Options propres à l'administrateur dans les commandes utilisateurs(UMS+FMS)

1. Dans la fonction 'vishnu_connect', un administrateur peut donner l'identifiant d'un utilisateur pour se connecter sous le nom de cet utilisateur dans VISHNU.

2. Dans la fonction 'vishnu_list_history_cmd', l'administrateur peut lister toutes les commandes de tout les utilisateurs ou les commandes d'un utilisateur en particulier en fournissant l'identifiant de l'utilisateur.

3. Dans la fonction 'vishnu_list_local_accounts', l'administrateur peut lister toutes les comptes de tout les utilisateurs ou les comptes d'un utilisateur particulier en fournissant l'identifiant de l'utilisateur.

4. Dans la fonction 'vishnu_list_options', l'administrateur peut lister les toutes les options de tout les utilisateurs ou les options d'un utilisateur en particulier en fournissant l'identifiant de l'utilisateur.

5. Dans la fonction 'vishnu_list_sessions', l'administrateur peut lister les toutes les sessions de tous les utilisateurs ou les sessions d'un utilisateur en particulier en fournissant l'identifiant de l'utilisateur, ou les sessions sur une machine particulière en fournissant l'identifiant de la machine.
6. Dans la fonction 'vishnu_list_file_transfers', l'administrateur peut lister tous les transferts de fichiers de tous les utilisateurs ou ceux d'un utilisateur en particulier en fournissant l'identifiant de l'utilisateur, ou lister les transferts impliquant une machine particulière (qui peut être source ou destination du transfert) en fournissant l'identifiant de la machine.
7. Dans la fonction 'vishnu_stop_file_transfers', l'administrateur peut annuler tous les transferts de fichiers de tous les utilisateurs ou ceux d'un utilisateur en particulier en fournissant l'identifiant de l'utilisateur, ou annuler les transferts impliquant une machine particulière (qui peut être source ou destination du transfert) en fournissant l'identifiant de la machine.

7.6 Gestion des processus VISHNU et délestage (IMS)

- L'administrateur peut lister les processus VISHNU, sur toute la plateforme ou sur une machine particulière. Fonction associée : vishnu_get_processes.
- L'administrateur peut arrêter un processus VISHNU, ce processus ne sera pas redémarrer automatiquement. Attention : l'administrateur doit avoir un compte sur la machine Fonction associée : vishnu_stop
- L'administrateur peut redémarrer un processus VISHNU sur une machine, ce processus doit avoir déjà tourné pour VISHNU sur cette machine et l'administrateur doit avoir un compte sur la machine. Fonction associée : vishnu_restart
- L'administrateur peut délester une machine selon deux modes. Dans le mode HARD, tout les processus VISHNU de la machine sont arretés. Dans le mode SOFT, seul FMS et TMS sont touchés, on arrête tout leurs transferts et jobs en cours. Fonction associée : vishnu_load_shed

7.7 Surveillance de l'état des machines (IMS)

- Un administrateur peut fixer la fréquence de mise à jour de l'enregistrement de l'état des machines. Fonction associée : vishnu_set_update_frequency
- Un administrateur peut obtenir la fréquence de mise à jour de l'enregistrement de l'état des machines. Fonction associée : vishnu_get_update_frequency
- Un administrateur peut fixer un seuil sur une machine. Ce seuil peut être l'utilisation du CPU, la mémoire libre restante ou l'espace disque restant. Lors de l'enregistrement de l'état d'une machine, si un seuil est atteint sur cette machine, l'administrateur est notifié par mail de ce dépassement. Fonction associée : vishnu_set_threshold
- Un administrateur peut obtenir les seuils fixés sur une machine. Pour plus d'informations sur les seuils voir la partie pour fixer les seuils ci-dessus. Fonction associée : vishnu_get_threshold

7.8 Définition des formats des identifiants (IMS)

- Un administrateur peut fixer le format des identifiants VISHNU automatiquement générés pour les utilisateurs, les machines, les jobs soumis aux batchs scheduler et les transferts de fichiers. Ces identifiants peuvent contenir plusieurs variables :
  - '$DAY' : Variable qui sera remplacée par le jour de création (1-31)
  - '$MONTH' : Variable qui sera remplacée par le mois de création (1-12)
  - '$YEAR' : Variable qui sera remplacé par l'année de création (0-99)
  - '$CPT' : Variable compteur qui est automatiquement incrémentée
  - '$SITE' : Pour les utilisateurs ou machine, une information sur le lieu
  - '$UNAME' : Pour les utilisateurs, variable remplacée par le nom de l'utilisateur

– '$MANAME' : Pour les machines, variable remplacée par le nom de la machine

Attention la variable conteur est obligatoire pour éviter la génération d'un identifiant déjà existant. Fonctions associées : 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` *userPasswordEncryption*] [`-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=VISHNU.listUsers(string sessionKey, ListUsers listuser, 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`] [`-s` *URI*] [`-n` *name*] *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.

-s *URI* The URI of the supervisor to control the processes.

-n *name* The path to the supervisord script file on 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:

"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] [-s URI] [-n name] 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.

-s *URI* The URI of the supervisor to control the processes.

-n *name* The path to the supervisord script file on 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:

"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`] [`-s` *URI*] [`-n` *name*] *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.

`-s` *`URI`* The URI of the supervisor to control the processes.

`-n` *`name`* The path to the supervisord script file on 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:

"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, const SupervisorOp& op = SupervisorOp());

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).

op Input argument. The option for the supervision.

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 string& machineId, const SupervisorOp& op = SupervisorOp());

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.

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

op Input argument. The option for the supervision.

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 SupervisorOp& op);

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.

op 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=VISHNU.getProcesses(string sessionKey, ListProcesses process, 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=VISHNU.getSystemThreshold(string sessionKey, ListThreshold value, 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, SupervisorOp op = SupervisorOp());

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).

op Input argument. The option for the supervision.

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, string machineId, SupervisorOp op = SupervisorOp());

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.

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

op Input argument. The option for the supervision.

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, SupervisorOp op);

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.

op 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])