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, Ibrahima Cissé, Rodrigue Chakode, Benjamin Depardon, Haïkel Guémar, and Amine Bsila	June 13, 2013	

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

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME
9	24/06/2011	Ajout de l'option vishnuMachineId dans les fichiers de configuration de UMS, IMS et FMS .	I.CISSE
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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
22	27/08/2012	Nouvelle compilation avec les batchs	K. COULOMB
23	01/10/2012	Ajout des prérequis en termes de connaissances systèmes et raffinement de la procédure d'installation	R. Chakode
24	08/11/2012	Ajout d'un paragraphe sur l'utilisation d'une base de données distante	K. COULOMB
25	08/11/2012	Prise en compte du gestionnaire de resources PBSPro	A. BSILA
26	21/11/2012	Première MAJ pour ZeroMQ	K. Coulomb
27	04/01/2013	Maj documentation du dispatcher	B. Depardon
28	15/01/2013	Ajout d'une FAQ pour la résolution des problèmes d'administration. Ajout d'une entrée dans la FAQ à propos du chargement des plugins TMS	H. Guémar
29	15/01/2013	Mise à jour des dépendences d'installation avec ZeroMQ et suppression du chapite lié aux web services	R. Chakode
30	28/01/2013	Présentation des tests automatiques	A. Bsila
31	26/02/2013	Ajout documentation pour le support des gestionnaires de cloud	R. Chakode
32	14/05/2013	Ajout documentation pour le chiffrement des connexions à la base de données MySQL	R. Chakode
33	16/05/2013	Ajout documentation pour le chiffrement des connexions à la base de données PostGreSQL	R. Chakode
34	13/06/2013	Ajout documentation pour le chiffrement des communications entre les clients et les SeDs	R. Chakode

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

1.4 Documents de 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.

Chapter 2

Acronymes et Glossaire

2.1 Terminologies

- Sysfera-DS : Désigne la suite logiciel développée par SysFera pour simplifier et l'utilisation des ressources de calcul à grande échelle. La suite comprend VISHNU et le WebBoard.
- VISHNU est une boite à outil qui permet la fédération et l'accès unifié à des plateformes de calcul distribués (clusters, grilles et cloud).
- FMS (File Management Service): désigne le système de gestion de données dans VISHNU.
- IMS (Information Management Service): désigne le système de gestion d'information dans VISHNU.
- TMS : Task Management Service ou système de gestion de tâches
- UMS : User Management Service ou système de gestion des utilisateurs
- SeD FMS : désigne l'exécutable qui supporte le service FMS.
- SeD IMS : désigne l'exécutable qui supporte le service VISHNU.
- SeD TMS : désigne l'exécutable qui supporte le service TMS.
- SeD UMS : désigne l'exécutable qui supporte le service UMS.
- Client FMS : désigne l'ensemble des programmes pour accéder aux services FMS.
- Client IMS : désigne l'ensemble des programmes pour accéder aux services IMS.
- Client TMS : désigne l'ensemble des programmes qui permettent d'accéder aux services offerts par le Sed TMS.
- Client UMS : désigne l'ensemble pour accéder aux services UMS.
- Dispatcher: est un composant de VISHNU ayant pour rôle de simplifier la configuration des clients dans des achitectures complexes. Comme décrit dans le document d'architecture [ARCH], il sert de broker entre les clients et différents batch schedulers dont il rend l'accès transparent pour les utilisateurs.
- Supervisord est un outil de supervision externe que l'on peut intégrer dans l'environnement de déploiement pour contrôler le démarrage, l'arrêt et le redemarrage automatique des services.
- 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.
- ZeroMQ aussi appelé ZMQ est une bibliothèque de communication par passage de messages asynchrones utilisé dans VISHNU pour assurer la communication entre les différents composants (clients, SeD et Dispatcher).
- Préfrontale : c'est une machine mise en amont des frontales des serveurs de calcul.

Chapter 3

Installation à partir des sources

Ce chapitre décrit l'installation de VISHNU à partir des sources.

3.1 Prérequis

La compilation nécessite les bibliothèques suivantes :

- **Dépendances obligatoires**
  - 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 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
  - ZeroMQ 2.x. Non supporté avec les versions 3.x
  - libuuid : est une dépendance de ZeroMQ
  - GLIBC 2.7 ou ultérieure
  - OpenSSH 4.2 ou ultérieure
  - Moteur de base de données :
    * PostGreSQL 8.0 minimum, ou
    * MySQL 5.1 minimum
- **Dépendances spécifiques pour le module UMS**
  - OpenLDAP 2.4 ou ultérieure
- **Dépendances spécifiques pour IMS**
  - Sigar 1.6.4 disponible sur sourceforge (http://sourceforge.net/projects/sigar/files/sigar/1.6/)
  - Supervisord 3.0.a, installable à partir des outils Python (ex. pip install supervisor)
  - Xmlrpc-c 2.1.19
- **Dépendances spécifiques aux batch schedulers**
  Sauf indication contraire, seules les versions indiquées sont officiellement supportées car ayant fait l'objet de tests.
  - Torque 2.3.6

  - IBM LoadLeveler 2.x ou 3.x
  - SLURM 2.2.x, 2.3.x ou 2.4.x
  - LSF 7.0.6.134609
  - Grid Engine 2011.11
  - PBSPro 10.4
  - Detalcloud 1.x. Le support de Deltacloud nécessite la bibliothèque Libdeltacloud Client pour C et C++, version 0.9 ou supérieure.
- **Dépendences nécessaires à la génération des APIs Python et Java**
  - SWIG 1.3.40 (ATTENTION : la version 2 de SWIG n'est pas supporté).
  - JAVA SDK 1.6.
  - Python 2.6

3.2 Compilation des sources

VISHNU utilise CMake comme système de construction. Les différentes fonctionnalités sont activées ou désactivées à la demande grâce à des variables Cmake spécifiques. Positionnée à *ON* une variable permet d'activer la fonctionnalité associée. Ex. -DCOMPILE_UMS=ON active la compilation du module UMS. Un positionnement à *OFF* désactive la compilation de la fonctionnalité. Ex. -DCOMPILE_UMS=ON désactive la compilation du module UMS.

Dans certains cas on peut avoir besoin de mots-clés spécifiques pour activer une fonctionnalité. Ex. -DVISHNU_BATCH=SLURM permet de sélectionner SLURM comme batch scheduler sous-jacent à TMS.

La liste ci-dessous décrit les variables disponibles :

- COMPILE_CLIENT_CLI permet, si on le souhaite, de compiler les binaires clients en ligne de commande. Par défault seules les libs sont compilées.
- COMPILE_SERVERS permet si on le souhaite de compiler les serveurs. Par défault ils ne sont pas compilés.
- CMAKE_INSTALL_PREFIX définit le répertoire d'installation. Par défault l'installation est réalisée dans */usr/local*.
- COMPILE_UMS indique si le module UMS sera compilé.

  Ce module est activé par défault (-DCOMPILE_UMS=ON).
- COMPILE_FMS spécifie si le module FMS sera compilé.

  FMS est désactivé par défaut (-DCOMPILE_FMS=OFF), lorsqu'il est activé le module COMPILE_UMS dont il dépend doit être également activé (-DCOMPILE_UMS=ON).
- COMPILE_TMS permet de préciser si le module TMS sera compilé.

  Par défaut désactivé, si TMS est activé (-DCOMPILE_TMS=ON), alors les modules UMS et FMS dont il dépend doivent être égalements activés (-DCOMPILE_UMS=ON et -DCOMPILE_FMS=ON).

  Par ailleurs lorsque TMS est activé, on doit obligatoirement sélectionner un batch scheduler grâce aux options VISHNU_BATCH et VISHNU_BATCH_VERSION (ex. VISHNU_BATCH=SLURM et VISHNU_BATCH_VERSION=2.2).
- COMPILE_IMS indique si le module IMS sera compilé.

  IMS est désactivé par défaut. Lorsqu'il est activé, tous les autres modules doivent l'être également (-DCOMPILE_UMS=ON, -DCOMPILE_TMS=ON et -DCOMPILE_FMS=ON).
- ENABLE_PYTHON : Désactivée par défaut cette option permet d'activer ou non la compilation du code PYTHON.
- ENABLE_JAVA : Désactivée par défaut cette option permet d'activer ou non la compilation du code JAVA.
- ENABLE_LDAP permet d'activer le support LDAP pour l'authentification. Cette option est désactivée par défaut.

- ENABLE_SWIG permet d'activer la generation du code des adapteurs PYTHON et JAVA.

  Désactivée par défaut cette option doit obligatoirement être activée si on choisit de ne pas compiler tous les modules VISHNU. C'est-à-dire lorsqu'au moins l'une des options COMPILE_UMS, COMPILE_TMS, COMPILE_FMS ou COMPILE_IMS est à OFF. Par ailleurs, l'activation de cette option nécessite que SWIG ait été installé au préalable.

- VISHNU_BATCH indique le batch scheduler à activer (TORQUE par défaut).

  Selon l'installation de votre batch scheduler et la configuration de votre système, il peut être nécessaire de positionner manuellement des variables cmake supplémentaires pour faciliter la recherche des fichier d'entête et les bibliothèques dynamiques ou statiques nécessaires à la compilation et l'édition de liens. Leurs valeurs doivent être des chemins absolus. Ainsi:

  - Pour le scheduler Tivoli LoadLeveler, LOADLEVELER_INCLUDE_DIR doit pointer vers le repertoire contenant le fichier llapi.h tandis que LOADLEVELER_LIB doit pointer vers le fichier libllapi.so ou le fichier le libllapi.a.
  - Pour TORQUE, la variable TORQUE_INCLUDE_DIR doit pointer vers le dossier contenant le fichier pbs_ifl.h tandis que TORQUE_LIB doit pointer vers le fichier libtorque.so ou le fichier libtorque.a.
  - Pour SLURM, la variable SLURM_INCLUDE_DIR doit pointer vers le repertoire contenant slurm/slurm.h tandis que SLURM_LIB doit pointer vers le chemin absolu du libslurm.so ou le fichier libslurm.a.
  - Pour LSF, les variables LSF_INCLUDE_DIR, LSBATCH_LIB et LSBATCH_LIB doivent être définies. La variable LSF_INCLUDE doit pointer vers le chemin absolu du repertoire contenant le fichier lsf/lsbatch.h, LSBATCH_LIB doit pointer vers le chemin absolu du fichier libbat.so ou du fichier libbat.a. Tandis que la variable LSBATCH_LIB doit pointer vers la bibliothèque libbat.so ou libbat.so.
  - Pour les Batch schedulers basés sur Grid Engine, les variables SGE_ROOT, SGE_INCLUDE_DIR, SGE_BIN_DIR, SGE_BIN_DIR et SGE_LIB doivent être défines. La valeur de SGE_ROOT doit à la racine du dossier d'installation du batch scheduler; SGE_INCLUDE_DIR doit pointer vers le dossier contenant le fichier d'entête drmaa.h; SGE_BIN_DIR doit pointer vers le repertoire contenant les fichiers binaires. tandis que SGE_LIB doit pointer vers la bibliothèque libdrmaa.so ou libdrmaa.a.
  - Pour PBS PRO, les variables PBS_INCLUDE_DIR et PBS_LIB doivent être défines comme suit. PBS_INCLUDE_DIR doit pointer vers le repertoire contenant le fichier d'entête pbs_ifl.h, tandis que PBS_LIB doit pointer vers la bibliothèque libpbs.so ou libpbs.a.
  - Pour Deltacloud, les variables LIBDELTACLOUD_INCLUDE_DIR et LIBDELTACLOUD_LIB doivent être défines comme suit. LIBDELTACLOUD_INCLUDE_DIR doit pointer vers le repertoire contenant les fichiers d'entête de LibDeltacloud, tandis que LIBDELTACLOUD_LIB doit pointer vers la bibliothèque libdeltacloud.so ou libdeltacloud.a.

- VISHNU_BATCH_VERSION indique la version du batch scheduler utilisé

- BUILD_TESTING: Désactivée par défaut, cette variable spécifie si le module de test sera compilé.

Ci-dessous nous décrirons les étapes pour installer les clients et les serveurs d'UMS et TMS dans /opt/vishnu. TORQUE est utilisé comme backend à TMS. La compilation des APIs Python sera également activée.

- 1. Créer un répertoire build à la racine du projet et s'y placer

  $ *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
    -DCOMPILE_CLIENT_CLI=ON
    -DCOMPILE_SERVERS=ON

  - 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 *
    -DCOMPILE_CLIENT_CLI=ON
    -DCOMPILE_SERVERS=ON
    -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

3.3 Vérifier l'installation

En supposant que la variable d'environnement INSTALL_PREFIX pointe à la racine du repertoire d'installation (/opt/vishnu dans notre exemple), vous devez avoir les programmes suivants en fonction des modules installés:

- $INSTALL_PREFIX/sbin/dispatcher : binaire du Dispatcher
- $INSTALL_PREFIX/sbin/umssed : binaire du SeD UMS
- $INSTALL_PREFIX/sbin/fmssed : binaire du SeD FMS
- $INSTALL_PREFIX/sbin/tmssed : binaire du SeD TMS
- $INSTALL_PREFIX/sbin/imssed : binaire du SeD IMS
- $INSTALL_PREFIX/bin : contient les clients ligne de commande dont les noms sont préfixés par *vishnu_*.
  Ex. vishnu_connect est le client ligne de commande pour se connecter à VISHNU.
- $INSTALL_PREFIX/lib : contient les librairies dynamiques nommées sous la forme *libvishnu**.
- $INSTALL_PREFIX/share : contient les pages de manuel pour les différents programmes ligne de commande.
- $INSTALL_PREFIX/etc : contient des exemples de fichiers de configuration.

Chapter 4

Configuration de la base de données

Les fichiers de configuration de la base de données 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, référez-vous 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 à créer les tables de la base de données tandis que le second sert à initialiser les données de la base.

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, référez-vous 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 -h localhost vishnu ;*
- Se connecter à la base de données
  $ *psql -h localhost vishnu ;*
- Créer le schéma de la base de données
  $ \i */path/to/postgre_create.sql*
- Initialiser la base de données
  $ \i */path/to/database_init.sql*

4.3 Bases de données distantes et Parefeu

VISHNU repose sur une base de données. Les serveurs de VISHNU ont besoin de s'y connecter pour fonctionner. Lorsqu'il n'y a aucune restriction sur les ports accessibles entre différentes machines (cas fréquent dans un réseau local), il n'y a généralement pas de problème pour se connecter à une base de données distante en spécifiant correctement le databaseHost dans les fichiers de configuration des éléments de VISHNU. Dans le cas où il y a des restrictions sur les ports ouverts, ou dans le cas de réseaux complexes avec du NAT par exemple, il est souvent plus facile de passer par des tunnels ssh pour encapsuler les connexions vers la base de données.

Supposons par exemple que la base de données soit sur une machine A et que le SeD VISHNU se trouve sur une machine B. Supposons aussi que le port d'écoute de la base de données est 3306. Ainsi, selon la machine à partir de laquelle le tunnel est crée, nous pouvons avoir deux cas de figure.

- Si le tunnel est ouvert depuis la machine A vers la machine B, on parlera de tunnel "reverse". A partir de la machine A, il suffit s'exécuter une commande de la forme : "reverse") :
  ssh -t -t -R 3306:localhost:3306 user@B
- Si le tunnel est ouvert depuis la machine B vers la machine A, on parlera de tunnel "forward". A partir de la machine B, il suffit s'exécuter une commande de la forme :
  ssh -t -t -L 3306:localhost:3306 user@A

Pour tester que le tunnel fonctionne, placez-vous sur la machine B et connectez-vous à MySQL :

```
mysql –u vishnu_user –h 127.0.0.1 –p
```

Après avoir tapé le mot de passe, vous devriez vous retrouver connecté sur la base de données sur la machine A. Il ne vous reste plus qu'à configurer le fichier de configuration de l'élément VISHNU avec les éléments suivants (à adapter en fonction de votre configuration) :

```
databaseType=mysql
databaseHost=127.0.0.1
databaseName=vishnu
databaseUserName=vishnu_user
databaseUserPassword=vishnu_user
databaseConnectionsNb=5
```

Attention : si vous utilisez localhost comme nom de machine pour vous connecter à la base de données, vous risquez de rencontrer des problèmes si mysql est installé sur B. En effet, mysql tente de se connecter en utilisant une socket linux (généralement /var/run/mysqld/mysqld.sock), au lieu de tenter de se connecter à une machine distante. Utilisez bien 127.0.0.1 pour databaseHost.

4.4 Utilisation de LDAP

Nous supposons que vous avez déjà une installation de LDAP opérationnelle. Si ce n'est pas le cas, référer vous 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 ENABLE_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

Configuration et démarrage des services

5.1 Résumé des programmes serveurs

Selon les modules sélectionnés, les binaires suivants doivent être disponibles sur votre système après l'installation:

- Dispatcher: $INSTALL_PREFIX/sbin/dispatcher
- SeD UMS: $INSTALL_PREFIX/sbin/umssed
- SeD FMS: $INSTALL_PREFIX/sbin/fmssed
- SeD TMS : $INSTALL_PREFIX/sbin/tmssed
- SeD IMS : $INSTALL_PREFIX/sbin/imssed

5.2 Configuration des SeDs

Cette section décrit les clés de configuration des différents composants de VISHNU. Pour une première configuration, vous pouvez vous inspirer du fichier d'exemple *vishnu-sample.cfg* situé dans le dossier $INSTALL_PREFIX/etc et également décrit ci-dessous. Tout texte après un dièse '#' est un commentaire.

```
# This is a commented sample configuration file for VISHNU
# Copyright (c) SysFera SA
# Last update: 16/01/2013
#

# Legends:
#  (M): Indicates that a key is mandatory for all components and MUST not be empty.
#  (M<List of Components>): Indicates a key is specific and mandatory for the
#                           listed components.
#  (O): Indicates that a key is optional.
#  (O)<List of Components>: Indicates that a key is optional only for the
#                           listed components.
#  (OS<List of Components>): Indicates a key is optional and specific to
#                             the listed components.
#   E.g. (OS<TMS>) means that a key is optional and specific to the Task
#   Management System (TMS) module.
#
# Definitions
# FMS: File Management System
# IMS: Information Management System
```

```
# TMS: Task Management System
# UMS: User Management System
# FQDN: Full Qualified Domain Name


##############################################################################
#               Common Parameters                                            #
##############################################################################
# vishnuId (M<FMS,IMS,TMS,UMS>): Sets the identifier of the VISHNU instance
#
vishnuId=1

# databaseType (M<FMS,IMS,TMS,UMS>): Defines the type of the database.
# Possible values are 'mysql' or 'postgresql'
#
databaseType=mysql

# databaseHost (M<FMS,IMS,TMS,UMS>): Defines the IP address or the FQDN of the
# database server.
# Here we assume that the database and VISHNU are hosted on the same server
#
databaseHost=localhost

# databaseName (M<FMS,IMS,TMS,UMS>): Sets the name of the database instance
#
databaseName=vishnu

# databaseUserName (M<FMS,IMS,TMS,UMS>): Sets the login name for authenticating
# against the database
#
databaseUserName=vishnu

# databaseUserPassword (M<FMS,IMS,TMS,UMS>): Sets the password associated to
# the database user
#
databaseUserPassword=vishnu


##############################################################################
#                Dispatcher Related Parameters                               #
##############################################################################


# disp_uriAddr (M<Dispatcher,Client>):
#  * For Dispatcher this corresponds to the address on which it'll listen on
#    for client requests
#  * For Clients this indicates the address for connecting to the Dispacther
#
disp_uriAddr=tcp://127.0.0.1:5560

# disp_uriSubs (M<Dispatcher>|O<IMS|TMS|FMS|UMS>):
# ** For the Dispatcher, it indicates the address to listen on for SeD subscription
# ** For SeD (FMS, IMS, TMS, UMS), this corresponds to the address from which
#    the module will register itself to the Dispatcher
#
disp_uriSubs=tcp://127.0.0.1:5561

# disp_nbthread (OS<Dispatcher>):
# Sets the number of workers threads in the Dispatcher
#
# In a platform with a high number of concurrent request, increase
# the number of workers may be interesting for reducing response time.
```

```
# Conversely, if the number of concurrent requests is low, decrease the may
# preserve useless resource consumption.
#
disp_nbthread=5


##############################################################################
#                UMS Related Parameters                                      #
##############################################################################

# ums_uriAddr (M<UMS>|O<Dispatcher,Client>):
#   * For UMS, this sets the address and the port on which the UMS SeD
#     will listen on
#   * For Dispactcher and clients it indicates address(es) for connecting
#     to UMS SeDs.
#     For Dispatcher this should correspond to a list of colon-seperated pair
#     in the form of: ums_uriAddr=uri_umssed1 sed1_name;uri_umssed2 sed2_name;...
#     uri_umssed<i> correspond to valid URIs. sed<i>_name set the
#     names associated to the SeDs
#     E.g. ums_uriAddr=tcp://127.0.0.1:5562 localsed;tcp://192.168.1.1:5562 lansed
#
ums_uriAddr=tcp://127.0.0.1:5562

# sendmailScriptPath (M<UMS>): Sets the path to the script for sending emails
#
sendmailScriptPath=/opt/software/GIT/vishnu_1/core/src/utils/sendmail.py

# authenticationType (M<UMS>): Defines the underlying authentication mode
# VISHNU supports four modes of authentication:
#  * UMS: Authentication against the built-in user management system (default).
#  * LDAP: Authentication against an LDAP directory
#  * UMSLDAP: Use both modes, looking first in the native authentication database
#  * LDAPUMS: Use both modes, looking first in the LDAP directory
# If not set, UMS mode will be selected.
#
#authenticationType=UMS


##############################################################################
#                FMS Specfic Parameters                                      #
##############################################################################

# fms_uriAddr (M<FMS>|O<Dispatcher,Client>):
#   * For FMS, this sets the address and the port on which the FMS SeD
#     will listen on
#   * For Dispactcher and clients it indicates address(es) for connecting
#     to FMS SeDs.
#     For Dispatcher this should correspond to a list of colon-seperated pair
#     in the form of: fms_uriAddr=uri_fmssed1 sed1_name;uri_fmssed2 sed2_name;...
#     uri_fmssed<i> correspond to valid URIs. sed<i>_name set the
#     names associated to the SeDs
#     E.g. fms_uriAddr=tcp://127.0.0.1:5563 localsed;tcp://192.168.1.1:5563 lansed
#
fms_uriAddr=tcp://127.0.0.1:5563


##############################################################################
#                IMS Related Parameters                                      #
##############################################################################

# ims_uriAddr (M<IMS>|O<Dispatcher,Client>):
#   * For IMS, this sets the address and the port on which the IMS SeD
```

```
#     will listen on
#   * For Dispactcher and clients it indicates address(es) for connecting
#     to IMS SeDs.
#     For Dispatcher this should correspond to a list of colon-seperated pair
#     in the form of: ims_uriAddr=uri_imssed1 sed1_name;uri_imssed2 sed2_name;...
#     uri_imssed<i> should correspond to valid URIs. sed<i>_name set the
#     names associated to the SeDs.
#     E.g. ims_uriAddr=tcp://127.0.0.1:5564 localsed;tcp://192.168.1.1:5564 lansed
#
ims_uriAddr=tcp://127.0.0.1:5564


################################################################################
#                TMS Related Parameters                                        #
################################################################################
# tms_uriAddr (M<TMS>|O<Dispatcher,Client>):
#   * For TMS, this sets the address and the port on which the TMS SeD
#     will listen on
#   * For Dispactcher and clients it indicates address(es) for connecting
#     to TMS SeDs.
#     For Dispatcher this should correspond to a list of colon-seperated pair
#     in the form of: ims_uriAddr=uri_tmssed1 sed1_name;uri_tmssed2 sed2_name;...
#     uri_tmssed<i> correspond to valid  URIs. sed<i>_name set the
#     names associated to the SeDs.
#     E.g. tms_uriAddr=tcp://127.0.0.1:5565 localsed;tcp://192.168.1.1:5565 lansed
#
tms_uriAddr=tcp://127.0.0.1:5565

# batchSchedulerType (M<TMS>): Defines the type of the batch scheduler TMS
# will handle.
# VISHNU supports TORQUE, LOADLEVELER, SLURM, LSF, SGE and PBS
batchSchedulerType=SLURM

# intervalMonitor (M<TMS>): In seconds, this key defines the interval after
# which the jobs are monitored
#
intervalMonitor=30

# defaultBatchConfig (OS<TMS>): Sets the path to the default batch configuration
# file.
#
#defaultBatchConfig=$HOME/defaultbatch.cfg


################################################################################
#                Other Parameters                                              #
################################################################################

# vishnuMachineId (M<FMS,IMS,TMS>|O<UMS>): Optional for UMS and mandatory for FMS,
# IMS and TMS, this key corresponds to the identifier of the machine in VISHNU
# environment.
# When set, it MUST correspond to valid machine identifier.
#
vishnuMachineId=machine_1

# timeout (M<Dispatcher>|O<FMS,IMS,TMS,UMS>): In seconds, this defines the
# duration afer which a request is considered as expired.
#
timeout=120

#urlSupervisor (M<FMS,IMS,TMS,UMS>): Specifies the address of Supervisord
urlSupervisor=http://127.0.0.1:9001
```

5.2.1 Configuration de Supervisod

Nous vous recommandons de lancer les SeDs via Supervisord. C'est un moniteur d'exécution qui contrôle l'exécution des services en offrant des fonctionnalités de resilience. En effet, lorsqu'un service tombe de manière inattendue, il se charge de le redemarrer de manière transparente.

L'exemple ci-dessous décrit la configuration de Supervisord. Il est inspiré du fichier d'exemple *supervisord-sample.cfg* situé dans le dossier $INSTALL_PREFIX/etc. Vous pouvez vous en inspirer pour l'adapter à votre installation. Tout texte après un point-virgule ';' est un commentaire.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;; PARAMETRES LIES A LA PRISE EN CHARGE DES COMPOSANTS DE VISHNU ;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Attention: il faut bien respecter les nommages 'umssed', 'imssed',
; 'tmssed', 'fmssed'. Sinon ils ne seront pas bien pris en charge

; Définir le SeD UMS comme un processus à surveiller
; Son alias est umssed
; command indique la commande à exécuter. A modifier selon votre installation
[program:umssed]
command=umssed ~/conf/ums_config.cfg ; Commande comme on ferait dans un terminal

; Définir le SeD FMS comme un processus à surveiller
; Son alias est fmssed
; command indique la commande à exécuter. A modifier selon votre installation
[program:fmssed]
command=fmssed ~/conf/fms_config.cfg

; Définir le SeD IMS comme un processus à surveiller
; Son alias est imssed
; command indique la commande à exécuter. A modifier selon votre installation
[program:imssed]
command=imssed ~/conf/ims_config.cfg

; Définir le SeD TMS comme un processus à surveiller
; Son alias est tmssed
; command indique la commande à exécuter. A modifier selon votre installation
[program:tmssed]
command=tmssed ~/conf/tms_config.cfg

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;; PARAMETRES SPECIFIQUES A SUPERVISORD ;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Définir l'URL de connexion à Supervisord
[supervisorctl]
serverurl=http://127.0.0.1:9001

; inet (TCP) server disabled by default
; ip_address:port specifier, *:port for all if
[inet_http_server]
port=127.0.0.1:9001
```

5.2.2 Notes spécifiques pour l'intégration TMS-DELTACLOUD

TMS est une composante clé d'une architecture VISHNU. Grâce à son architecture souple, il s'intègre dans différents types d'environnements de calcul. Des environnements traditionnels reposant sur des batch schedulers classiques comme LoadLeveler

et SLURM, il a été étendu pour s'interfacer avec les gestionnaires de gestionnaires de ressources virtuelles (aussi appelés gestionnaires de cloud) tels tels qu'OpenStack et OpenNebula. Basé sur Delatcloud et actuellement en version beta, cette extension n'est actuellement testée qu'avec OpenStack.

L'intégration avec les gestionnaires de ressources virtuelles nécessite un mode de fonctionnement interne relativement différent de ce qui se passe avec les batch schedulers traditionnels. En effet, étant donné que ces gestionnaires ne gèrent pas les tâches, VISHNU fournit des fonctionnalités supplémentaires pour assurer cette gestion. Conceptuellement, chaque tâche est exécutée au sein d'une machine virtuelle qui est automatiquement instanciée à cet effet. Cette machine virtuelle est également automatiquement détruite à la fin de tâche. Par ailleurs, pour simplifier la gestion des données d'entrée et de sortie, VISHNU permet de définir un système de fichiers NFS qui est monté automatiquement dans l'arborescence de la machine virtuelle au démarrage. De manière résumée, pour gérer le cycle de vie des tâches, VISHNU assure entre autres :

- L'authentifiation sur le gestionnaire de ressources virtuelles.
- L'instanciation et le démarrage des machines virtuelles.
- Le démarrage des scripts au sein des machines virtuelles.
- La supervision des scripts et des machines virtuelles
- La suspension des machines virtuelles lorsque les tâches qu'elles exécutent sont terminées.

La prise en charge de ces fonctionnalités nécessite un ensemble de paramètres de configuration supplémentaires. Ces paramètres sont singuliers pour la gestion de ressources virtuelles et ne s'appliquant pas aux batch schedulers traditionnels. C'est pour cela qu'il sont définis en dehors du fichier de configuration principal pour mieux gérer leur caractère optionnel tout en simplifiant la configuration de l'ensemble. Deux étapes simples sont nécessaires pour configurer l'intégration avec Deltacloud. Dans un premier temps, vous devez spécifier dans le fichier de configuration principal que le batch scheduler sous-jacent est Deltacloud (*batchSchedulerType=DELTACLOUD*). Et, ensuite, définir les paramètres spécifiques pour la gestion de ressources virtuelles.

Ces paramètres, listés ci-dessous, sont définis au niveau global par l'administrateur et peuvent etre surpassés par des options spécifiques définies par l'utilisateur lors de la soumission d'une tâche. Voir le manuel utilisateur pour plus de détails:

- Les variables d'environnements.

  Définies sur la machine où est lancé le SeD TMS, ces variables sont :

  - VISHNU_CLOUD_ENDPOINT : Définit l'url d'accès à l'API Delatcloud.

    Ex. *VISHNU_CLOUD_ENDPOINT=http://192.168.1.1:3001/api* indique que l'API Delatcloud est accessible via le port 3001 de la machine ayant l'adresse IP 192.168.1.1, à noter le contexte /api qui est obligatoire. Ceci suppose que sur cette machine *deltacloudd* a été démarrée pour écouter sur ce port.
  - VISHNU_CLOUD_USER : Définit la chaine de connexion au gestionnaire de ressources virtuelles. En général cela correspond au login de l'utilisateur. Cependant, pour OpenStack, ceci est une chaine sous la forme *user+tenant*. Où *user* et *tenant* désignent respectivement le login de l'utilisateur et le nom du tenant auquel il appartient.

    Ex. *VISHNU_CLOUD_USER=bob+sysfera* définit la chaine de connexion pour l'utilisateur bob du tenant sysfera.
  - VISHNU_CLOUD_USER_PASSWORD : Définit le mot de passe à associer à la chaine de connexion.
  - VISHNU_CLOUD_VM_IMAGE : Définit l'identifiant de l'image à utiliser pour instancier les machines virtuelles. Cette valeur doit être valide et correspondre à une image que l'utilisateur défini par VISHNU_CLOUD_USER a le droit d'instancier.
  - VISHNU_CLOUD_VM_USER : Définit l'identifiant de l'utilisateur sous lequel les tâches seront exécutées au sein des machines virtuelles. Cet identifiant n'est pas crée par VISHNU, il doit donc correspondre à un identifiant réel au sein de la machine virtuelle.
  - VISHNU_CLOUD_VM_USER_KEY : Ce paramètre définit le nom de la clé SSH à déployer dans les machines virtuelles. La valeur doit correspondre à une clé enregistrée dans le gestionnaire de cloud.
  - VISHNU_CLOUD_DEFAULT_FLAVOR : Définit la flavor par défaut des machines virtuelles. Idem, cette valeur doit correspondre à une flavor définie dans le gestionnaire de cloud.
  - VISHNU_CLOUD_NFS_SERVER : Définit l'adresse du serveur NFS sur lequel sont stockées les données des applications s'exécutant au sein des machines virtuelles.
  - VISHNU_CLOUD_NFS_MOUNT_POINT : Désigne le point de montage sur le serveur NFS.

- Le fichier *.vishnurc*

  Si définit dans le dossier personnel de l'utilisateur sous lequel s'exécute le SeD TMS, ce fichier est lu à chaque soumission de tâche pour surpasser les valeurs des paramètres définies via les variables d'environnement. Dans ce fichier, les paramètres sont définis sous-forme *clé=valeur*, avec une entrée par ligne. Voir l'exemple ci-dessous. Les lignes commençant par dièse '#' sont des commentaires.

5.3 Déploiement dans un même sous-réseau

Dans cette section nous supposons que tous les composants vont être déployés sur une même machine nommée **prefrontale**.

NOTES : 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.

Une fois la configuration des différents composants terminée, suivez attentivement les étapes suivantes pour les déployer:

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

   Dans cette commande, on demande au dispatcher de démarrer en utilisant les serveurs définis dans le fichier 'config'. Si le fichier de configuration n'est pas fourni, des options par défaut sont utilisées. Les différents options de configurations et leur valeur par défaut sont décrites deux sections plus loin.
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).

5.4 Le cas multi-réseaux

Le cas multi-réseaux ne change pas du cas précédent, les étapes sont toutes les mêmes. Le seul ajout à faire est d'ajouter des tunnels SSH entre les machines serveurs et la machine ou se trouve la base de données s'il n'y a pas d'accès direct sur le port correspondant au port de la base de données, et un second tunnel pour l'enregistrement des serveurs sur le dispatcher. Par la suite, il est conseillé d'utiliser le dispatcher pour que le client n'ait plus qu'à se connecter au dispatcher pour réaliser ses requêtes (le dispatcher jouant alors un rôle de proxy entre le client et les réseaux).

Les tunnels SSH sont faits avec une commande similaire à l'une des suivantes:

- ssh -fN -L 8889:localhost:8889 machine_distance

  Etablir un tunnel entre la machine locale et la machine distante au travers du port 8889.
- Etablir un reverse tunnel entre la machine locale et la machine distante au travers du port 8889 ssh -R -fN 8889:localhost:8889 machine_distante

5.5 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)`

5.6 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.

5.7 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.

5.8 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.

5.9 Les scripts de démarage automatique

Des scripts de démarage sont préparés pour pouvoir exécuter automatiquement les serveurs VISHNU. Les scripts sont installés lors de l'installation des serveurs dans install/etc/ (ou dans /etc si c'est installé dans /usr).

Chapter 6

Configuration avancées et sécurité

6.1 TMS en mode natif

Après installation, il est possible d'utiliser les fonctionnalités de TMS sur une machine ne possédant pas de batch scheduler. Ce mode de fonctionnement natif est appelé TMS Posix Shell. Le type de batch scheduler associé est 'POSIX'.

Lors de la compilation, ce module est activé automatiquement et indépendamment du batch scheduler sélectionné. Il est également possible de ne compiler que ce module en choisissant 'POSIX' comme bacth scheduler lors de la compilation. Le flag dans le fichier de configuration du serveur TMS sera également POSIX. Il faut noter que lors de la compilation de VISHNU pour un batch scheduler classique, ce pseudo batch est automatiquement compilé, et l'utilisateur peut lors du submit explicitement demander à soumettre sur le batch posix en utilisant l'option -p. Sinon, c'est systématiquement le vrai batch scheduler auquel vishnu soumet. Cela implique que d'un point de vue système, vishnu a constamment 3 processus 'tmssed' dans le cas d'un batch scheduler classique, un premier qui est le serveur réalisant les services, et les 2 autres correspondent à des moniteurs utilisés par vishnu (pour interroger les batchs schedulers sous-jacents) pour avoir les données à jour associées aux jobs. De plus, afin de gérer les jobs utilisateurs et pouvoir les annuler, pour chaque utilisateur ayant un job, cet utilisateur aura un démon qui tournera, pourra controller les jobs, et fournir des informations au moniteur au travers d'une socket unix.

6.2 Connexions en mode sécurisé

Par défaut, les communications entre les clients VISHNU et les SeD, de même que les communications entre les SeD et la base de données sont non chiffrées. C'est-à-dire qu'elles transitent en clair sur le réseau. Ceci peut constituer un facteur de vulnérabilité. Afin d'améliorer la sécurité, VISHNU intègre depuis les versions ultérieures à la version 3.0.0, le support de chiffrement des communications (clients--SeDs, SeDs -- base de données). Ce chiffrement se base sur des certificats SSL, et repose sur la librairie OpenSSL http://www.openssl.org/. Vous pouvez consulter la documentation officielle pour plus d'information sur le chiffrement et les certificats http://www.openssl.org/docs/.

Non activé par défaut, vous devez explicitement activer le support SSL. Avant toute chose, vous devez *disposer d'un certificat valide*; vous pouvez utiliser un certificat signé par une autorité de certification ou un certificat autosigné (voir la documentation d'OpenSSL). Pour les plus pressés, le script *create-ssl-cert.sh* fourni avec VISHNU permet de créer rapidement des certificats autosignés. Avant de l'exécuter, vous devez l'éditez pour adapter le mot de passe du certificat, ainsi que les paramètres liés au nom du serveur pour lequel le certificat sera généré.

Cela se fait à deux niveaux : au niveau des communications entre les SeDs et les clients et au niveau des communications avec les SeDs et la base de données. La prochaine section décrit comment activer le support SSL entre les SeD et les clients VISHNU, tandis que la section suivante décrira la configuration des connexions SSL entre les SeDs et la base de données.

6.3 Activation du support SSL coté client et coté SeDs

Dans un premier temps, vous devez avoir compilé VISHNU avec le support SSL. C'est normalement le cas par défaut. Ensuite, vous devez activer des clés de configuration appropriées selon que vous êtes du coté client ou coté SeDs :

- `useSsl=1` : de type entier et nécessaire aussi bien coté client que coté SeDs, cette clé doit avoir une valeur non nulle pour que le support SSL soit pris en compte.
- `serverSslCertificate=/path/to/server/certificate` : requis coté SeDs si l'option `useSsl` est activée, cette clé indique le chemin vers le certificat.
- `serverPrivateKey=/path/to/server/private/key` : requis coté SeDs si l'option `useSsl` est activée, cette clé indique le chemin vers la clé privée du serveur.
- `sslCa=/path/to/cafile` : optionnel coté client, cette clé indique le chemin vers le fichier contenant la liste des certificats de l'autorité de certification. Ce paramètre est notamment nécessaire pour un certificat autosigné.

L'activation ou la mise à jour d'un ou plusieurs de ces paramètres coté SeD nécessite le redémarrage des services.

6.4 Chiffrement des connexions à la base de données

Aussi bien MySQl que PostGreSQL disposent d'un support natif pour l'utilisation de connexions SSL.

6.4.1 Connexions SSL vers une base de données MySQL

Les étapes de configuration décrites ci-dessous sont tirées de la documentation officielle de MySQL disponible à l'adresse suivante : http://dev.mysql.com/doc/refman/5.0/en/ssl-connections.html. Les étapes de la procédure comportent :

- La vérification de la disponibilité du support SSL dans MySQL
- La génération d'un certificat autosigné (optionnel)
- L'activation du support SSL sur le serveur MySQL
- Le test de connexion en SSL

6.4.1.1 Vérifier que MySQL a été compilé avec le support SSL

Les binaires et les librairies client et serveur MySQL doivent avoir été compilés avec le support d'OpenSSL. Si ce n'est pas votre cas, vous devez réinstaller MySQL. Les étapes suivantes indiquent comment vérifier la disponibilité du support d'OpenSSL :

- Coté serveur, exécuter la commande suivante : `$ mysqld --ssl --help`
  Si une erreur se produit, alors le binaire n'a pas été compilé avec le support SSL.
- Coté client, exécuter la commande suivante : `$ mysql --ssl --help`
  Si une erreur se produit, alors le binaire n'a pas été compilé avec le support SSL.

6.4.1.2 Activer le support SSL dans le serveur MySQL

Pour activer le support SSL sur le serveur MySQL, vous devez indiquer un certificat, la clé privé du serveur et les certificats de l'autorité de confiance (optionnel, mais requis si le certificat est autosigné). Le certificat peut être autosigné ou signé par une autotité de certification.

Une fois que vous disposez des différentes entités du certificat, la configuration de MySQL est assez simple :

- Editez le fichier de configuration de MySQL et indiquer les entités du certificat dans la section *[mysqld]*.

```
[mysqld]
ssl-ca=/opt/etc/sysfera/mysql-ssl/ca.pem
ssl-cert=/opt/etc/sysfera/mysql-ssl/server-cert.pem
ssl-key=/opt/etc/sysfera/mysql-ssl/server-key.pem
```

Adapter les chemins en fonction de l'emplacement où vous avez stocké les différentes entités. Le paramètre *ssl-ca* doit pointer vers l'emplacement où vous avez stocké le fichier des certificats de l'autorité de confiance, *ssl-cert* doit pointer vers l'emplacement où vous avez stocké le certificat du serveur, tandis que le paramètre *ssl-key* doit pointer vers l'emplacement où vous avez stocké la clé privé du serveur.

- Redemarrer MySQL et vérifier que le support SSL a correctement été activé :
  - Lancez un terminal et connectez-vous à MySQL en tant que super-utilisateur (root) : `$ mysql -u root -p`
  - Vérifier que le module SSL a été correctement chargé : `mysql> show variables with '%ssl%'`
    Si le support SSL a été correctement activé, vous devez avoir une sortie comme celle affichée ci-dessous :

```
+---------------+----------------------------------------+
| Variable_name | Value                                  |
+---------------+----------------------------------------+
| have_ssl      | YES                                    |
| ssl_ca        | /opt/etc/sysfera/mysql-ssl/ca.pem      |
| ssl_cert      | /opt/etc/sysfera/mysql-ssl/server-cert.pem |
| ssl_key       | /opt/etc/sysfera/mysql-ssl/server-key.pem  |
+---------------+----------------------------------------+
```

Les valeurs des paramètres ssl_ca, ssl_cert et ssl_key doivent être conformes à votre configuration.

6.4.1.3 Tester la connexion en SSL à MySQL

La connexion en SSL à une base de données MySQL nécessite que l'utilisateur MySQL ait dispose des privilèges spécifiques (`REQUIRE SSL`). Ce privilège est activé lors de l'attribution des rôles à l'utilisateur par le biais de la commande `GRANT`.

Exemple :

```
mysql> GRANT SELECT ON vishnu.*
       TO 'secure_user'@'127.0.0.1'
       IDENTIFIED BY 'password' REQUIRE SSL;
```

Cette commande indique que l'utilisateur 'secure_user' ne pourra se connecter à la base de donnée 'vishnu' qu'en mode SSL. Si l'option `REQUIRE SSL` est omise, seul le mode de connexion par défaut sera activé. Même si le support SSL a été activé au niveau du serveur.

Pour se connecter à la base de données, il suffit d'exécuter la commande suivante. L'option *--ssl-ca* n'est requise que si le certificat a été autosigné ou a été généré par une autorité de certification non reconnue.

```
$ mysql -u secure_user -p \
  --ssl-ca=/opt/etc/sysfera/mysql-ssl/ca.pem
```

6.4.2 Connexion SSL vers une base de données PostGreSQL

Les étapes de configuration décrit dans cette section sont tirées de la documentation officielle de PostGreSQL disponible ici http://docs.postgresqlfr.org/8.3/libpq-ssl.html (pour le client), et ici http://docs.postgresqlfr.org/8.3/ssl-tcp.html (pour le serveur). Les étapes de la procédure comportent :

- La vérification de la disponibilité du support d'OpenSSL dans PostGreSQL
- La génération de certificats autosignés (optionnel)
- L'activation du support SSL au niveau du serveur PostGreSQL
- Le test de connexion en SSL

6.4.2.1 Vérifier que PostGreSQL a été compilé avec le support SSL

Coté serveur, vous devez vérifier que le binaire *postmaster* est lié à la bibliothèque d'OpenSSL (*libssl*). Pour le faire, vous pouvez utiliser la commande suivante à partir d'une console.

```
$ ldd /path/to/postmaster  | grep ssl
```

De même, coté client, assurer vous que la bibliothèque *libpq* est liée à *libssl*. Pour le faire, vous pouvez utiliser la commande suivante à partir d'une console.

```
$ ldd /path/to/libpq.so  | grep ssl
```

Si toutes les vérifications sont correctes, vous pouvez passer à la configuration. Sinon vous devez reinstaller PostGreSQL en activant le support d'OpenSSL lors de la compilation (`./configure --with-openssl`).

6.4.2.2 Activer le support SSL dans le serveur PostGreSQL

L'activation du support SSL dans PostGreSQL est assez simple :

Premièrement, éditez le fichier de configuration (`PostgreSQL.conf`)et positionnez le paramètre `ssl` à `on`

```
ssl=on
```

Ensuite :

- Le certificat du serveur doit être nommé `server.crt` et placé dans le repertoire de données (*data*) de PostGreSQL. A noter que le fichier doit appartenir à l'utilisateur *postgres* et avoir les droits d'accès `600`.
- La clé privé du certificat doit être nommée `server.key` et placée dans le repertoire de données (*data*) de PostGreSQL. Il doit également appartenir à l'utilisateur *postgres* et avoir les droits d'accès `600`.
- Le fichier contenant les certificats de l'autorité doit est être nommé `root.crt` et placé dans le repertoire de données (*data*) de PostGreSQL. Il doit appartenir à l'utilisateur *postgres* et avoir les droits d'accès `644`.

Si vous souhaitez avoir plus de détails ou mettre en oeuvre des configurations avancées comme la prise en charge des certificats revoqués, vous pouvez consulter la documentation officielle disponible à cette adresse http://docs.postgresqlfr.org/8.3/ssl-tcp.html.

Une fois les configurations terminées, redémarrez le serveur PostGreSQL.

6.4.2.3 Tester la connexion en SSL à PostGreSQL

Contrairement à MySQL, vous n'avez pas besoin d'avoir des droits spécifiques pour se connecter en SSL. Il vous suffit :

- Premièrement, de copier le fichier contenant les certificats de l'autorité dans l'emplacement `~/.postgresql/root.crt` depuis le dossier personnel de l'utilisateur. En notant que ce fichier doit avoir les droits d'accès `644`.
- Ensuite, d'ajouter l'option "`sslmode=verify-ca`" lorsque vous vous connectez à PostGreSQL. Voir l'exemple ci-dessous.

```
$ psql -U postgres "sslmode=verify-ca"
```

Si tout se passe bien, alors vous pouvez terminer la configuration en activant les options nécessaires dans VISHNU.

6.4.3 Configurer VISHNU pour se connecter en SSL à une base de données

Une fois MySQL ou PostGreSQL a été configuré pour fonctionner en SSL, il suffit d'ajouter les lignes suivantes dans le fichier de configuration de VISHNU pour indiquer les paramètres de connexion en SSL.

```
databaseUseSsl=1
databaseSslCa=/opt/etc/sysfera/mysql-ssl/ca.pem
```

Si le paramètre *databaseUseSsl* a une valeur non-nulle, on indique à VISHNU de se connecter à la base de données en mode chiffré. Optionnel, le paramètre*databaseSslCa* indique l'emplacement des certificats de l'autorité de confiance. Il n'est utile que dans le cas des certificats autosignés ou signés par une autorité de certification non reconnue. Pour PostGreSQL, l'option *databaseSslCa* peut ne pas être indiquée. Dans ce cas, le chemin par défaut (`~/.postgresql/root.crt`) sera considéré.

Une fois la configuration terminée, redémarrez les services. Les communications avec la base de données se feront dorénavant en mode chiffré.

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 compteur 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.

7.9 FAQ

- Si vous rencontrez des problèmes lors du chargement des plugins TMS, vous pouvez configurez la variable d'environnement *LD_LIBRARY_PATH* pour pointer sur le répertoire contenant les plugins.

Chapter 8

Tests

Les tests automatiques permettent de valider les fonctionnalités de VISHNU et de garantir la non régression lors des développements. Le module de test peut être activé au moment de la construction de VISHNU, en activant la variable Cmake BUILD_TESTING. Le module de test sera donc compilé avec les sources de vishnu.

8.1 Tests unitaires

un certain nombre de tests unitaires sont fournis avec VISHNU. Lorsque vous compilez VISHNU avec BUILD_TESTING, ces tests sont construits et présents dans le sous-répertoire bin de votre répertoire de build. Les tests unitaires ont tous un nom terminant par UnitTests. Pour les lancer, il suffit juste d'appeler le binaire. Pour voir les options qu'il est possible de leur passer, vous pouvez lancer le binaire avec l'option --help (ce sont les options du framework boost tests).

Ces tests ne nécessitent pas d'avoir une plateforme VISHNU déployée.

8.2 Configuration des tests fonctionnels

Lorsque vous compilez Vishnu avec BUILD_TESTING, ces tests sont construits et présents dans le sous-répertoire bin de votre répertoire de build. Les tests fonctionnels ont tous un nom qui suit la nomenclature suivante: <nom du module>_automTest<numéro> (e.g., fms_automTest11 pour la suite de test 11 du module FMS). Pour les lancer, il suffit juste d'appeler le binaire. Pour voir les options qu'il est possible de leur passer, vous pouvez lancer le binaire avec l'option --help (ce sont les options du framework boost tests)

Ces tests, contrairement aux tests unitaires, nécessitent d'avoir une plateforme VISHNU déployée. L'administrateur doit fournir un ensemble de paramètres d'exécutions liés à la plateforme de test. Il est nécessaire de positionner 2 variables d'environnement.

8.2.1 Variables d'environnement et options de configuration

- VISHNU_CLIENT_TEST_CONFIG_FILE : Variable qui contient le chemin vers le fichier de configuration client. Ce fichier permet au client VISHNU de se connecter à la plateforme (voir le manuel utilisateur pour plus d'informations sur la configuration de ce fichier).
- VISHNU_TEST_SETUP_FILE : Variable qui contient le chemin vers le fichier de données de description de la plateforme. Ce fichier permet aux tests automatiques de s'exécuter sur la plateforme. Ce fichier doit contenir un ensemble de paramètres :
  - TEST_UMS_AUTHEN_TYPE : Contient le type de système d'authentification de la plateforme de test;
  - TEST_VISHNU_MACHINEID1 : Contient l'identifiant d'une machine VISHNU sur laquelle les tests UMS vont être exécutés;
  - TEST_WORKING_DIR : Contient le chemin vers le répertoire de travail local des tests;

- TEST_ROOT_VISHNU_LOGIN : Contient le login VISHNU de l'utilisateur ROOT de la plateforme. L'utilisateur ROOT a les droits administrateur et a un Local Account sur chacune des machines de test;
- TEST_ROOT_VISHNU_PWD : Contient le mot de passe VISHNU de l'utilisateur ROOT de la plateforme;
- TEST_ADMIN_VISHNU_LOGIN : Contient le login VISHNU de l'administrateur de la plateforme. L'administrateur est un utilisateur qui a les droits administrateur et qui n'a aucun Local Account sur aucune des machines de test;
- TEST_ADMIN_VISHNU_PWD : Contient le mot de passe VISHNU de l'administrateur de la plateforme;
- TEST_USER_VISHNU_LOGIN : Contient le login VISHNU de l'utilisateur de test de la plateforme. L'utilisateur doit avoir un Local Account sur chacune des machines de test. Tout les test FMS et TMS sont exécutés en tant que l'utilistaeur TEST_USER_VISHNU_LOGIN;
- TEST_USER_VISHNU_PWD : Contient le mot de passe VISHNU de l'utilisateur de test de la plateforme;
- TEST_FMS_HOST1 : Contient l'identifiant de la première machine VISHNU sur laquelle les tests FMS vont être exécutés;
- TEST_FMS_HOST2 : Contient l'identifiant de la deuxième machine VISHNU sur laquelle les tests FMS vont être exécutés;
- TEST_FMS_HOST1_WORKING_DIR : Contient le chemin vers le répertoire de travail distant des tests FMS sur la machine TEST_FMS_HOST1. Ce répertoire accessible en lecture et ecriture par le Local Account de l'utilisateur TEST_USER_VISHNU_L sur la machine TEST_FMS_HOST1;
- TEST_FMS_HOST2_WORKING_DIR : Contient le chemin vers le répertoire de travail distant des tests FMS sur la machine TEST_FMS_HOST2. Ce répertoire accessible en lecture et ecriture par le Local Account de l'utilisateur TEST_USER_VISHNU_L sur la machine TEST_FMS_HOST2;
- TEST_FMS_USER_LOCAL_GROUP : Contient le groupe UNIX du Local Account de l'utilisateur TEST_USER_VISHNU_LOGI sur la machine TEST_FMS_HOST1;
- TEST_TMS_MACHINE_IDS : Contient la liste des machines TMS sur lesquelles les tests TMS vont êtres exécutés. Cette liste est constituée des couples: "<identifiant de machine> <batch scheduler>", séparé par des point-virgule. Exemple "TEST_TMS_MACHINE_IDS=machine_1 SLURM;machine_2 TORQUE"

8.2.2 Exemple de fichier de configuration

```
# This is a sample setup file for VISHNU tests
# Copyright (c) SysFera SA
# Last update: 04/02/2013
#


# Definitions
# FMS: File Management System
# IMS: Information Management System
# TMS: Task Management System
# UMS: User Management System


##############################################################################
#               Common Parameters                                            #
##############################################################################
TEST_VISHNU_MACHINEID1=machine_1
TEST_WORKING_DIR=/home/absila/machine_1
TEST_ROOT_VISHNU_LOGIN=root
TEST_ROOT_VISHNU_PWD=vishnu_user
TEST_ADMIN_VISHNU_LOGIN=admin_1
TEST_ADMIN_VISHNU_PWD=admin
TEST_USER_VISHNU_LOGIN=user_1
TEST_USER_VISHNU_PWD=toto


##############################################################################
#                UMS Related Parameters                                      #
```

```
################################################################################
TEST_UMS_AUTHEN_TYPE=UMS

################################################################################
#                 FMS Specfic Parameters                                       #
################################################################################
TEST_FMS_HOST1=machine_1
TEST_FMS_HOST2=machine_1
TEST_FMS_HOST1_WORKING_DIR=/home/absila/machine_2
TEST_FMS_HOST2_WORKING_DIR=/home/absila/machine_3
TEST_FMS_USER_LOCAL_GROUP=test;

################################################################################
#                 TMS Related Parameters                                       #
################################################################################
TEST_TMS_MACHINE_IDS=machine_1 SLURM;machine_2 TORQUE
```

8.2.3 Prérequis

Afin que les tests automatiques s'exécutent, la plateforme Vishnu doit être fonctionnelle et les variable d'environnement renseignées. Tous les paramètres du fichier VISHNU_TEST_SETUP_FILE doivent être renseignés.

Les différentes machines de test peuvent être une seule et même machine (l'exemple de fichier ci-dessus fait référence à une seule machine).

8.2.4 Rapport de test

Les tests automatiques génèrent des rapports résumant le déroulement du test ainsi que les résultats. Ces rapports peuvent être générés en XML, pour cela il faut exécuter la commande suivante : "make test_name-xml" en remplaçant test_name par le nom du test à exécuter (e.g., fms_automTest1). Ce format XML est pratique pour intégrer les résultats de test dans votre environnement d'intégration continue tel que Jenkins.

Chapter 9

UMS Command reference

9.1 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

9.2 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), 2 (DELETED).

ENVIRONMENT

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

DIAGNOSTICS

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

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

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

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

"Vishnu not available (System)" [4]

"Internal Error: Undefined exception" [9]

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

"Missing parameters" [14]

"Vishnu initialization failed" [15]

"Undefined error" [16]

"The userId is unknown" [21]

"The user is locked" [23]

"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

9.3 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

9.4 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

9.5 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

9.6 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

9.7 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"

9.8 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), 2 (DELETED).

-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

9.9 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

9.10 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

9.11 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. The current supported options are VISHNU_CLOSE_POLICY (how to close the session), VISHNU_TIMEOUT (timeout before closing a session automatically), VISHNU_TRANSFER_CMD (to use rsync or scp in FMS)

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

9.12 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 0 0 -b uid=\$USERNAME,ou=users,c

9.13 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), 2 (DELETED).

-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

9.14 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 10

UMS C++ API Reference

10.1 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]

10.2 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]

10.3 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]

10.4 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]

10.5 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]

10.6 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]

10.7 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]

10.8 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]

10.9 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]

10.10 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]

10.11 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. The current supported options are VISHNU_CLOSE_POLICY (how to close the session), VISHNU_TIMEOUT (timeout before closing a session automatically), VISHNU_TRANSFER_CMD (to use rsync or scp in FMS)

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]

10.12 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]

10.13 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]

10.14 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 11

UMS Python API Reference

11.1 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])

11.2 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])

11.3 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])

11.4 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])

11.5 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])

11.6 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])

11.7 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])

11.8 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])

11.9 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])

11.10 VISHNU.listUsers

VISHNU.listUsers — lists VISHNU users

Synopsis

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

DESCRIPTION

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

ARGUMENTS

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

listuser Output argument. Listuser is the list of users .

options Input argument. Allows an admin to get information about a specific user identified by his/her userId or to get information about users authenticated by a specific user-authentication system.

RETURNED OBJECTS

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

EXCEPTIONS

The following exceptions may be thrown:

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

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

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

SystemException("Vishnu not available (System)" [4])

SystemException("Internal Error: Undefined exception" [9])

UMSVishnuException("The userId is unknown" [21])

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

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

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

11.11 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. The current supported options are VISHNU_CLOSE_POLICY (how to close the session), VISHNU_TIMEOUT (timeout before closing a session automatically), VISHNU_TRANSFER_CMD (to use rsync or scp in FMS)

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])

11.12 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])

11.13 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])

11.14 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 12

IMS Command reference

12.1 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

12.2 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

12.3 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

12.4 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

12.5 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

12.6 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

12.7 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

12.8 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

12.9 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

12.10 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

12.11 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

12.12 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

12.13 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

12.14 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 13

IMS C++ API Reference

13.1 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]

13.2 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]

13.3 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]

13.4 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]

13.5 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]

13.6 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]

13.7 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]

13.8 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]

13.9 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]

13.10 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]

13.11 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]

13.12 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]

13.13 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]

13.14 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 14

IMS Python API Reference

14.1 VISHNU.getProcesses

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

Synopsis

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

DESCRIPTION

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

ARGUMENTS

sessionKey Input argument. The session key.

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

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

RETURNED OBJECTS

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

EXCEPTIONS

The following exceptions may be thrown:

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

14.2 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])

14.3 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])

14.4 VISHNU.getSystemThreshold

VISHNU.getSystemThreshold — gets a system threshold on a machine

Synopsis

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

DESCRIPTION

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

ARGUMENTS

sessionKey Input argument. The session key.

value Output argument. The thresholds value.

options Input argument. The options for the threshold.

RETURNED OBJECTS

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

EXCEPTIONS

The following exceptions may be thrown:

SystemException("The database generated an error" [2])

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

14.5 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])

14.6 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])

14.7 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])

14.8 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])

14.9 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])

14.10 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])

14.11 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])

14.12 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])

14.13 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])

14.14 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])