Waarp r66 Documentation

Version 3.6.0-1

Waarp SAS

août 18, 2021

Table des matières

Version 3.6.0

Date août 18, 2021

Waarp R66 est une solution de tranfert de fichiers monitoré. Vous trouverez dans le présent document les informations nécéssaires à l'installation, la configuration, l'utilisation et l'administration des moniteurs (clients et serveurs) Waarp R66.

CHAPITRE 1

Démarage

1.1 Installation

1.1.1 Pré-requis

Les prérequis pour WaarpR66 sont les suivants :

- Un OS supporté : Linux (toutes distributions) et Windows 32 ou 64 bits
- Java 1.6 minimum (java 1.8 recommandé)
- Base de données PostgreSQL (version 9.4 minimum) est recommandée, MySQL, MariaDB, Oracle SQL server et H2 sont supportés.
- Les interfaces web nécessitent un navigateur récent (Chrome, Firefox, Internet Explorer 10+).
- Mémoire vive :
  - Pour un client : 128Mo de RAM minimum ; 512Mo recommandés
  - Pour un serveur : 512Mo de RAM minimum ; au moins 1Go recommandés. L'utilisation dépend de la charge du service. Par exemple, un serveur Waarp R66 a besoininstallation.rst d'au moins 2Go pour traiter 5 000 transferts simultanés.

  Par ailleurs, ces valeurs ne prennent en compte que les besoins des services R66. Si des transferts lancent des processus externes dans les chaînes de traitement (tâches *EXEC*), les besoins en RAM de ces processus viennent en sus.

Le chemin vers le dossier contenant Java peut être renseigné dans la variable d'environnement `JAVA_HOME` (ex : `export JAVA_HOME=/usr/lib/jvm/java8` ou `set JAVA_HOME="C:\Java"`).

Important : La variable d'environnement `JAVA_HOME` doit être définie pour les systèmes Windows.

1.1.2 Linux

Avec les packages systèmes

Des packages RPM sont fournis pour RHEL 6 et 7 (et Centos/Scientific Linux).

Prérequis : - le package java-1.7.0-openjdk doit être installé au préalable

Télécharger la dernière version du fichier RPM correspondant à la version de votre système d'exploitation depuis la page de téléchargement.

Installer les RPM avec les commandes :

```
rpm -i waarp-r66-server-[version].rpm
```

Vous pouvez ensuite passer à la configuration.

Avec les dépôts Waarp

Pour faciliter l'installation et les mises à jour de WaarpR66, nous fournissons des dépôts pour RHEL 6 (et Centos/Scientific Linux).

Pour ajouter les dépôts Waarp à votre système, suivez la procédure indiquée sur notre page de téléchargement.

Après avoir suivi cette procédure, vous pouvez installer WaarpR66 avec la commande :

```
yum install waarp-r66-server
```

Vous pouvez ensuite passer à la configuration.

Avec les archives autonomes

L'archive est autonome et fonctionne sur toutes les distributions linux. Il suffit de la décompresser et de suivre la procédure de configuration pour pouvoir utiliser WaarpR66.

Télécharger la dernière version tar.gz pour linux depuis la page de téléchargement.

Décompressez ensuite l'archive :

```
tar -xf waarp-r66-[version]_linux.tar.bz2
```

Vous pouvez ensuite passer à la configuration.

1.1.3 Windows

Avec les archives autonomes

L'archive est autonome et fonctionne sur toutes les versions de Windows. Il suffit de la décompresser et de suivre la procédure de configuration pour pouvoir utiliser WaarpR66.

Télécharger la dernière version zip pour Windows depuis la page de téléchargement. Décompressez l'archive *waarp-r66-[version]_windows.zip* et passez à la configuration.

La section suivante détaille le contexte multi-instance de WaarpR66 ainsi que la création d'une de ces instances.

1.2 Creation d'instance

1.2.1 Principe généraux

Waarp R66 est une solution multi-instance. Une unique installation peut servir à plusieurs moniteurs.

Pour cela le dossier de configuration répertorie les configurations des différentes instances dans un dossier du nom de leur `HOSTID`.

Le dossier de configuration dépends du type d'installation de waarp :
- `/etc/waarp/conf.d` pour une installation via les rpms
- `etc/conf.d` à partir du dossier d'extraction pour les archives autoportantes

Dans ce document ce dossier est nommé `CONFDIR`.

Ainsi un serveur portant 3 moniteurs (server1, server2, server3) Waarp R66 aurait une arborsence semblable à celle ci-dessous.
- *CONFDIR*/server1
- *CONFDIR*/server2
- *CONFDIR*/server3

1.2.2 Création d'une instance Linux

Création de la configuration

Pour facililter l'initialisation d'une instance, des modèles de configuration sont fournis. Ces modèles sont situés dans le dossiers `{TEMPLATES}` :
- `/usr/share/waarp/templates/` pour une installation via les rpms
- `share/templates` à partir du dossier d'extraction pour les archives autoportantes

Pour créer une instance, copiez le dossier de modèle dans le dossier correspondant à la configuration de l'instance :

```
cp -r {TEMPLATES} {CONFDIR}/$HOSTID
```

Ces fichiers sont préconfigurés pour une utilisation standard. Une partie de la configuration est dépendante de l'instance (identifiants, dossiers, etc.). Dans les fichiers XML de modèle, la chaîne `{{app_name}}` doit être remplacée par l'identifiant de l'instance :

```
for f in {CONFDIR}/$HOSTID/*.xml; do
   sed -i -r "s|{{app_name}}|$HOSTID|g" $f
done
```

Configuration de la base de données

Par défaut, la nouvelle instance est configurée pour utiliser la base de données embarquée H2. Pour utiliser une autre base de données, il faut la configurer dans les fichier *CONFDIR*/$HOSTID/server.xml et *CONFDIR*/$HOSTID/client.xml.

La configuration de la base de données se trouve dans le bloc XML <db>...</db> :

```
<db>
   <dbdriver>postgresql</dbdriver>
   <dbserver>jdbc:postgresql://localhost/waarp_r66</dbserver>
   <dbuser>waarp</dbuser>
   <dbpasswd>waarp</dbpasswd>
```

(suite sur la page suivante)

(suite de la page précédente)

```
   <dbcheck>false</dbcheck>
</db>
```

Initialisation de la base de données

Pour initialiser la base de données, exécuter la commande suivante :

```
# Avec les packages :
waarp-r66client $HOSTID initdb

# Avec les archives :
./bin/waarp-r66client.sh $HOSTID initdb
```

Démarrage du serveur

Si l'instance configurée est un serveur, vous pouvez mintenant le démarrer.

```
# Avec les packages :
waarp-r66server $HOSTID start

# Avec les archives :
./bin/waarp-r66server.sh $HOSTID start
```

1.2.3 Création d'une instance Windows

Création de la configuration

Pour facililter l'initialisation d'une instance, des modèles de configuration sont fournis. Ces modèles sont situés dans le dossier `share\templates` à partir du dossier d'extraction.

Pour créer une instance, copiez le dossier de modèle dans le dossier correspondant à la configuration de l'instance :

```
xcopy /S share\templates {CONFDIR}\%$HOSTID%
```

Ces fichiers sont préconfigurés pour une utilisation standard. Une partie de la configuration est dépendante de l'instance (identifiants, dossiers, etc.). Dans les fichiers XML de modèle, la chaîne `{{app_name}}` doit être remplacée par l'identifiant de l'instance %$HOSTID%.

Configuration de la base de données

Par défaut, la nouvelle instance est configurée pour utiliser la base de données embarquée H2. Pour utiliser une autre base de données, il faut la configurer dans les fichier *CONFDIR*`\HOSTID\server.xml` et *CONFDIR*`\HOSTID\client.xml`.

La configuration de la base de données se trouve dans le bloc XML `<db>...</db>` :

```
<db>
   <dbdriver>postgresql</dbdriver>
   <dbserver>jdbc:postgresql://localhost/waarp_r66</dbserver>
```

(suite sur la page suivante)

(suite de la page précédente)

```
   <dbuser>waarp</dbuser>
   <dbpasswd>waarp</dbpasswd>
   <dbcheck>false</dbcheck>
</db>
```

Initialisation de la base de données

Pour initialiser la base de données, exécuter la commande suivante :

```
bin\waarp-r66server.bat %HOSTID% initdb
```

Démarrage du serveur

Si l'instance configurée est un serveur, vous pouves mintenant le démarrer.

Pour une installation avec les archives, la commande est :

```
bin\waarp-r66server.bat %HOSTID% start
```

1.3 Mise à jour

1.3.1 Pré-requis

Les prérequis pour WaarpR66 sont les suivants :

- Un OS supporté : Linux (toutes distributions) et Windows 32 ou 64 bits
- Java 1.6 minimum (java 1.8 recommandé)
- Base de données PostgreSQL (version 9.4 minimum) est recommandée, MySQL, MariaDB, Oracle SQL server et H2 sont supportés.
- Les interfaces web nécessitent un navigateur récent (Chrome, Firefox, Internet Explorer 10+).
- Mémoire vive :
  - Pour un client : 128Mo de RAM minimum ; 512Mo recommandés
  - Pour un serveur : 512Mo de RAM minimum ; au moins 1Go recommandés. L'utilisation dépend de la charge du service. Par exemple, un serveur Waarp R66 a besoininstallation.rst d'au moins 2Go pour traiter 5 000 transferts simultanés.

  Par ailleurs, ces valeurs ne prennent en compte que les besoins des services R66. Si des transferts lancent des processus externes dans les chaînes de traitement (tâches *EXEC*), les besoins en RAM de ces processus viennent en sus.

Le chemin vers le dossier contenant Java peut être renseigné dans la variable d'environnement `JAVA_HOME` (ex : `export JAVA_HOME=/usr/lib/jvm/java8` ou `set JAVA_HOME="C:\Java"`).

Important : La variable d'environnement `JAVA_HOME` doit être définie pour les systèmes Windows.

1.3.2 Avec les packages

Pour une installation faite à partir des packages, utiliser une des commandes suivantes (selon la distribution) :

```
# Avec les dépôts
yum install waarp-r66

# avec le package rpm
yum install path/to/waarp-r66.rpm
```

Pour la mise à jour de la base, il est utile de lancer la commande suivante :

```
waarp-r66server {hostid} initdb -upgradeDb
```

1.3.3 Avec les archives autonomes

Pour une installation faite à partir les packages autonomes, la procédure est la suivante :

1. Si le serveur R66 ou filewatcher a été installé en tant que service, arrêter celui-ci. Pour Windows seulement : désinstaller le service.
2. Extraire l'archive au même niveau que l'ancienne installation.
3. Copier le contenu du dossier `etc` de l'ancienne installation vers le dossier `etc` de la nouvelle version.
4. Procéder de même avec le dossier data de l'ancienne installation.
5. Si le serveur R66 ou filewatcher a été installé en tant que service :
   — Pour windows seulement : réinstaller les services depuis la nouvelle installation.
   — Pour linux : mettre à jour les chemins du service avec les nouveaux dossiers.
6. Mettre à jour le schéma de la base (si elle est utilisée) :

```
waarp-r66server {hostid} initdb -upgradeDb
```

Enfin, redémarrer les services.

1.3.4 Avec les jars

Il est recommandé d'utiliser les jars `with-dependencies`.

Vous pouvez remplacer le jar en cours d'usage par un nouveau jar téléchargé.

Il est ensuite fortement recommande de procéder à la mise à jour de la base. Aucune perte de données, seul les schémas des tables et les index seront modifiés pour s'adapter à la dernière version, selon des mises à jours progressives, en fonction de la version installée (depuis la version `2.4.23`).

```
waarp-r66server {hostid} initdb -upgradeDb
```

1.4 Configuration Simple

1.4.1 Principes Généraux

Il existe 4 fichiers de configurations :

- client.xml, détail le fonctionnement du moniteur en mode client
- logback-client.xml, configuration des logs client
- server.xml, détail le fonctionnement du moniteur en mode serveur
- logback-server.xml, configuration des logs serveur

En plus de ces 4 fichiers 2 autres fichiers sont consommés par l'instance pour alimenter sa configuration en base de données (si applicable)

- authent.xml, détails d'authentification des moniteurs authorisés
- rules.xml, règles de transfert utilisable par le mooniteur

1.4.2 client.xml

Le tableau ci-dessous détail les groupes utilisés dans la configuration d'un client WaarpR66. La description de ces groupes est détaillée plus bas.

Balise	Status
identity	Obligatoire
ssl	Si utilisation de SSL
directory	Recommandé
db	Si utilisation d'une base de données
extendTaskFactory	Si nécessité d'ajouter des extensions de tâches

1.4.3 server.xml

Le tableau ci-dessous détail les groupes utilisés dans la configuration d'un serveur WaarpR66. La description de ces groupes est détaillée plus bas.

Balise	Description
identity	Obligatoire
ssl	Si utilisation de SSL
server	Obligatoire
network	Recommandé
directory	Recommandé
db	Obligatoire
extendTaskFactory	Si nécessité d'ajouter des extensions de tâches
pushMonitor	Si nécessaire pour un monitoring en mode PUSH REST Json vers un serveur tiers

Note : Pour les balises indiquées ci-dessus, les fichiers valeurs renseignées dans les fichiers server.xml et client.xml doivent être identiques. À défaut, deux instances distinctes seront configurées.

1.4.4 Détails

Identity

Le groupe <*identity*> des configurations client et serveur permet de définir l'identité du moniteur (hostids et mot de passe)

Balise	Description
identity	
hostid	Identifiant de l'instance utilisé pour les connections en clair
sslhostid	Identifiant de l'instance utilisé pour les connexions chiffrées
cryptokey	Chemin d'accès à la clef DES utilisée pour chiffrer les mots de passe

Server

Le groupe <*server*> de la configuration serveur permet de préciser les informations nécéssaire au fonctionement du serveur.

Balise	Description
server	
serveradmin	Login pour l'accès administrateur
serverpasswd	Mot de passe pour l'accès administrateur chiffré par <cryptokey>
usenossl	Le serveur accepte les connections non SSL
usessl	Le serveur accepte les connections ssl
usehttpcomp	Utilisation de la compréssion HTTP pour l'interface d'administration
uselocalexec	Utilisation du LocalExec R66 au lieu du
httpadmin	Chemin d'accès au serveur
admkeypath	Chemin d'accès au keystore pour l'authentification https
admkeystorepass	Mot de passe d'accès au keystore
admkeypass	Mot de passe d'accès à la clef

Network

Par defaut un serveur WaarpR66 utilise les ports suivants :
- 6668 : Communications R66 en clair
- 6669 : Communications R66 chiffrées
- 8066 : Interface web de suivi (désactivée par défaut)
- 8067 : Interface web d'administration (désactivée par défaut)
- 8088 : Interface REST des serveurs WaarpR66 autonomes (désactivée par défaut)

Cependant ces ports sont configurables via le groupe *network*.

Balise	Description
network	
serverport	Communications R66 en clair
serveraddresses	Adresses utilisées pour le protocole R66 (séparées par des virgules)
serversslport	Communications R66 chiffrées
serverssladdresses	Adresses utilisées pour le protocole R66 en SSL (séparées par des virgules)
serverhttpport	Interface web de suivi (désactivée par défaut)
serverhttpaddresses	Adresses utilisées pour l'interface web de supervision (séparées par des virgules)
serverhttpsport	Interface web d'administration (désactivée par défaut)
serverhttpsaddresses	Adresses utilisées pour l'interface web HTTPS d'administration (séparées par des virgules)
serverrestport	Interface REST des serveurs WaarpR66 autonomes (désactivée par défaut)

SSL

Afin d'utilisé des connexions chiffrées le groupe <*ssl*> doit etre configuré avec les catalogues de certificats authorisés et leurs mots de passe d'accès.

Balise	Description
ssl	
keypath	Chemin d'accès au keystore d'authentification de l'instance.
keystorepass	Mot de passe d'accès au keystore
keypass	Mot de passe d'accès à la clef
trustkeypath	Chemin d'accès au keystore des certificats de confiance de l'instance
trustkeystorepass	Mot de passe d'accès au trustkeystore
trustuseclientauthenticate	Si vrai, R66 n'acceptera que les clients authorisés via SSL

Directory

Le groupe <*directory*> permet de définir les dossiers utilisés par les moniteurs WaarpR66 pour l'émission et la réception de fichiers.

Balise	Description
directory	
serverhome	Dossier racine de WaarpR66. Les autres dossiers paramétrables sont définis relativement à celui-ci
in	Dossier de dépôt des fichiers reçus
out	Dossier où sont cherchés les fichiers à transférer
work	Dossier tampon où sont stockés les fichiers en cours de réception
arch	Dossier d'export XML de l'historique des transferts
conf	Dossier d'export XML de la configuration de l'instance

DB

WaarpR66 utilise une base de données pour stocker les informations nécessaires aux transferts (Moniteurs authorisés et règles de transferts). Le groupe <*db*> permet de configurer les accès à la base de données utilisé par le moniteur.

Balise	Description
db	
dbdriver	Driver JDBC à utiliser pour se connecter à la base de données (postgresql)
dbserver	URI JDBC de connection à la base de données (ex : jdbc:postgresql://localhost:5433/waarp)
dbuser	L'utilisateur à utiliser pour se connecter à la base de données
dbpasswd	Le mot de passe de l'utilisateur

Note : Il est possible de faire fonctionner les moniteurs sans base de données. Les fichiers *authent.xml* et *rules.xml* seront utilisés comme source de configuration.

ExtendTaskFactory

Nouveau dans la version 3.6.0 : Ajout du sous-ensemble `extendTaskFactory` qui contient l'option `extendedtaskfactories` : pour la Factory `org.waarp.openr66.s3.taskfactory.S3TaskFactory`, si la classe est dans le claspath, il n'est pas nécessaire de l'ajouter.

Le groupe <*extendTaskFactory*> permet de définir des Task Factories additionnelles pour étendre les capacités de R66.

Balise	Description
extendTaskFactory	
extendedtaskfactories	Liste (séparée par des virgules) des TaskFactory en tant qu'extension pour ajouter des tâches à WaarpR66 (implicite pour la Factory `org.waarp.openr66.s3.taskfactory.S3TaskFactory`, si la classe est dans le claspath).

PushMonitor

Cette section décrit comment monitorer R66 via des appels REST HTTP(s) vers un serveur tiers (en mode PUSH).

Nouveau dans la version 3.6.0 : Ajout du sous-ensemble `pushMonitor` qui contient les options communes `url`, `delay`, `intervalincluded`, `transformlongasstring`, `token`, `apiKey`, les options spécifiques``endpoint``, `keepconnection` et `basicAuthent` sont liées à une API REST en destination, les options spécifiques``index``, `prefix`, `username`, `paswd` et `compression` sont liées à Elasticsearch en destination.

Le groupe <*pushMonitor*> permet de définir les parammètres pour que le serveur R66 envoie son monitoring des transferts vers un serveur tiers en mode API REST Json.

Balise	Description
pushMonitor	
Partie commune	
url	URL de base pour les exports du moniteur en mode POST HTTP(S) JSON
delay	Délai entre deux vérifications de changement de statuts sur les transferts
intervalincluded	Si « True », les informations de l'intervalle utilisé seront fournies
transformlongasstring	Si « True », les nombres « long » seront convertis en chaîne de caractères, sinon ils seront numériques
token	Spécifie si nécessaire le token dans le cadre d'une authentification via Token
apiKey	Spécifie si nécessaire le password dans le cadre d'une authentification via ApiKey (format `apiId:apiKey`)
Partie API REST	
endpoint	End point à ajouter à l'URL de base
keepconnection	Si « True », la connexion HTTP(S) sera en Keep-Alive (pas de réouverture sauf si le serveur la ferme), sinon la connexion sera réinitialisée pour chaque appel
basicAuthent	Spécifie si nécessaire l'authentification basique
Partie Elasticsearch	
index	Contient le nom de l'index avec de possibles substitutions, dont `%%WARPHOST%%` pour le nom du host concerné, et les `%%DATETIME%%`, `%%DATEHOUR%%`, `%%DATE%%`, `%%YEARMONTH%%`, `%%YEAR%%` pour des substitutions de date et heure partiellement (`yyyy.MM.dd.HH.mm` à `yyyy`)
prefix	Spécifie si nécessaire un prefix global dans le cas d'usage d'un Proxy devant Elasticsearch
username	Spécifie si nécessaire le username (et son password) dans le cadre d'une authentification basique
paswd	Spécifie si nécessaire le password dans le cadre d'une authentification basique
compression	Spécifie si les flux sont compressés (par défaut True)

1.4.5 logback-{client,server}.xml

Les fichiers *logback*.xml* permettent de paramétrer les écritures de log. Veuillez vous référer au manuel en ligne de Logback pour configurer la façon dont les logs sont générés et écrits dans un fichier et/ou vers *syslog*.

Il est à noter qu'il est conseillé d'avoir les éléments suivants dans le fichier de configuration de Logback.

```
<configuration>
  <statusListener class="org.waarp.common.logging.PrintOnlyWarningLogbackStatusListener"␣
↪/>

  <appender name=... class=...><!-- Voir la documentation Logback -->
  </appender>

  <root level="warn">
    <appender-ref ref=... /><!-- Voir la documentation Logback -->
  </root>

  <logger name="ch.qos.logback" level="WARN"/>
  <logger name="org.apache.http" level="WARN"/>
  <logger name="io.netty" level="WARN"/>
  <logger name="io.netty.util.internal.PlatformDependent" level="DEBUG"/>
</configuration>
```

1.4.6 authent.xml

Le fichier d'authent permet de renseigner les paramètres de connections des instances WaarpR66. Ce fichier est consommé par la commande *loadauth* ou *loadconf* (voir utilisation). Une fois consommé ce fichier n'est plus utilisé (vous pouvez le mettre à jour pour le recharger plus tard).

Le fichier liste un moniteurs dans une balise <*entry*> détaillée ci-dessous. Ces balises sont regroupées au sein d'une balise <authent>.

Balise	Description
entry	
hostid	L'hostid du moniteur
address	Addresse ou entrée DNS du moniteur
port	Si le moniteur est un serveur, le port de destination
isssl	Le moniteur utilise SSL
admin	Le moniteur authorise les accès Admin via R66
isclient	Le moniteur n'est pas un serveur
key	Mot de passe du moniteur

Au minimum le fichier doit renseigner le moniteur qui l'utilise.

1.4.7 rules.xml

Les fichiers de règles permettent de détailler les règles utilisées par le moniteur ainsi que leur contenu. Ce fichier est consommé par la commande *loadauth* ou *loadrules* (voir utilisation). Une fois consommé ce fichier n'est plus utilisé (vous pouvez le mettre à jour pour le recharger plus tard).

Le fichier décrit une règle dans une balise <*rule*> détaillée ci-dessous. Ces balises sont regroupées au sein d'une balise <rules>.

Balise	Description
rule	
idrule	Nom de la règle
comment	Commentaire
hostids	Liste des moniteurs authorisés à utiliser la règle
mode	Le mode de la règle
rpretasks	Tâches executées par le receveur avant le transfert
rposttasks	Tâches executées par le receveur après le transfert
rerrortasks	Tâches executées par le receveur en cas d'erreur du transfert
spretasks	Tâches executées par l'envoyeur avant le transfert
sposttasks	Tâches executées par l'envoyeur après le transfert
serrortasks	Tâches executées par l'envoyeur en cas d'erreur du transfert

Les hostids de la balises hostids sont présentés comme suit :

```
<hostids>
  <hostid>hostid1</hostid>
  <hostid>hostid2</hostid>
</hostids>
```

Le mode de la règle peut etre un des suivant

- 1 : SEND, Envoie le fichier client -> serveur
- 2 : RECV, Demande le fichier serveur -> client
- 3 : SEND+MD5
- 4 : RECV+MD5
- 5 : SENDTHROUGHMODE
- 6 : RECVTHROUGHMODE
- 7 : SENDMD5THROUGHMODE
- 8 : RECVMD5THROUGHMODE

Les listes de tâches (rpretasks, rposttasks, rerrortasks, spretasks, sposttasks, serrortasks). sont présentées comme suit :

```
<rpretasks>
  <tasks>
    <task></task>
    <task></task>
    <task></task>
    ...
  </tasks>
</rpretasks>
```

Le contenue d'une balise <task> est détaillé ci-dessous :

Balise	Description
task	
type	Le type de tâche
path	Les options de cette tâche
delay	Temps (ms) accordé avant l'envoie d'un Time Out

1.4.8 Cryptographie

cryptokey

Cette clef DES est utilisée par les instances WaarpR66 pour chiffrer les mots de passe pour s'identifier sur les autres instances. Pour générer une nouvelle cryptokey :

```
$ cat /dev/urandom | head -c 8 > cryptokey.des
```

Pour régénérer le mot de passe {pwd} dans le fichier {output} avec la clé {key} :

```
./bin/waarp-password.sh -ki {key} -pwd {pwd} -po {output}
```

keystore

Le keystore contient la clef privée d'identification de l'instance WaarpR66 pour les communication SSL. Il s'agit d'un Java KeyStore de type keystore.

truststore

Le truststore contient les certificats des instances autorisés à communiquer via SSL avec l'instance WaarpR66. Il s'agit d'un Java KeyStore de type truststore.

adminstore

Le keystore contient la clef privée pour accéder à l'interface d'administration de l'instance WaarpR66 en https. Il s'agit d'un Java KeyStore de type keystore. Pour générer une nouveau keystore :

```
$ keytool -genkey -keyalg RSA -alias selfsigned -keystore keystore.jks -storepass␣
↪password -validity 360 -keysize 2048
```

Pour générer un nouveau truststore depuis un keystore existant

```
$ keytool -export -keystore keystore.jks -alias selfsigned -file cert.crt
$ keytool -import -alias selfsigned -file cert.crt -keystore truststore.jks
```

restsignkey

La clef REST est utilisée par Waarp Manager pour communiquer avec les serveurs Waarp afin de récupérer l'historiques des transferts. Pour générer une nouvelle clef de signature REST

```
$ cat /dev/urandom | head -c 64 > restsignkey.key
```

Attention : Dans le cadre d'une utiilisation de Waarp Manager, les clefs cryptokey et restsignkey doivent être partagé par toute les instances serveur WaarpR66 du parc et connu de Waarp Manager.

Les sections suivantes présentent :

1. Un exemple de fichier des configurations
2. Le détail complet des fichiers de configuration

La section d'après détails le lancement d'un serveur WaarpR66.

1.5 Exemple de Configuration

Les configurations suivantes sont des configurations minimales pour un serveur fonctionel.

1.5.1 client.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:x0="http://www.w3.org/2001/XMLSchema">
  <comment>Client configuration for server1</comment>
  <identity>
    <hostid>server1</hostid>
    <sslhostid></sslhostid>
    <cryptokey>etc/certs/cryptokey.des</cryptokey>
  </identity>
  <directory>
    <serverhome>.</serverhome>
    <in>./data/in</in>
    <out>./data/out</out>
    <arch>./temp/arch</arch>
    <work>./work/</work>
    <conf>./temp/conf</conf>
  </directory>
  <db>
    <dbdriver>postgresql</dbdriver>
    <dbserver>jdbc:postgresql://localhost:5432/server1</dbserver>
    <dbuser>waarp</dbuser>
    <dbpasswd>waarp</dbpasswd>
    <dbcheck>false</dbcheck>
  </db>
</config>
```

1.5.2 server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:x0="http://www.w3.org/2001/XMLSchema">
  <comment>Configuration file for a server with a Postgresql database</comment>
  <identity>
    <hostid>server1</hostid>
    <sslhostid></sslhostid>
    <cryptokey>etc/certs/cryptokey.des</cryptokey>
  </identity>
  <server>
    <serveradmin>admin</serveradmin>
    <serverpasswd>5a4b7c6a66065cbb622acefec8c3a302</serverpasswd>
    <usenossl>True</usenossl>
    <usessl>False</usessl>
    <usehttpcomp>False</usehttpcomp>
    <uselocalexec>False</uselocalexec>
    <httpadmin>share/admin-i18n</httpadmin>
    <admkeypath>etc/certs/adminkey.jks</admkeypath>
    <admkeystorepass>password</admkeystorepass>
    <admkeypass>password</admkeypass>
    <checkaddress>False</checkaddress>
    <checkclientaddress>False</checkclientaddress>
    <pastlimit>86400000</pastlimit>
    <minimaldelay>5000</minimaldelay>
    <digest>7</digest>
    <multiplemonitors>1</multiplemonitors>
  </server>
  <network>
    <serverport>6666</serverport>
    <serversslport>6667</serversslport>
    <serverhttpport>8066</serverhttpport>
    <serverhttpsport>8067</serverhttpsport>
  </network>
  <directory>
    <serverhome>.</serverhome>
    <in>./data/in</in>
    <out>./data/out</out>
    <arch>./temp/arch</arch>
    <work>./work/</work>
    <conf>./temp/conf</conf>
  </directory>
  <db>
    <dbdriver>postgresql</dbdriver>
    <dbserver>jdbc:postgresql://localhost:5432/server1</dbserver>
    <dbuser>waarp</dbuser>
    <dbpasswd>waarp</dbpasswd>
    <dbcheck>false</dbcheck>
  </db>
</config>
```

1.5.3 authent.xml

```
<authent>
  <entry>
    <hostid>server1</hostid>
    <address>127.0.0.1</address>
    <port>6666</port>
    <isssl>false</isssl>
    <key>password</key>
  </entry>
  <entry>
    <hostid>server2</hostid>
    <address>127.0.0.4</address>
    <port>6668</port>
    <isssl>false</isssl>
    <key>password</key>
  </entry>
</authent>
```

1.5.4 rule.xml

```
<rules>
  <rule>
    <idrule>defaut</idrule>
    <comment>The default transfer rule</comment>
    <hostids>
      <hostid>server1</hostid>
      <hostid>server2</hostid>
    </hostids>
    <mode>1</mode>
    <rpretasks>
      <tasks></tasks>
    </rpretasks>
    <rposttasks>
      <tasks></tasks>
    </rposttasks>
    <rerrortasks>
      <tasks></tasks>
    </rerrortasks>
    <spretasks>
      <tasks></tasks>
    </spretasks>
    <sposttasks>
      <tasks>
        <task>
          <type>DELETE</type>
          <path></path>
          <delay>0</delay>
        </task>
      </tasks>
    </sposttasks>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <serrortasks>
      <tasks></tasks>
    </serrortasks>
  </rule>
</rules>
```

1.6 Initialisation

1.6.1 Initialiser l'instance

Une fois l'instance {hostid} créée et configurée les commandes suivantes permettent d'initialiser et de charger sa configuration dans la base de données (fichier authent.xml et rules.xml).

```
#Créer le schéma de la base de données spécifiée dans server.xml
waarp-r66server {hostid} initdb
#Charge les données des fichiers authent.xml et rules.xml dans la base de donnés
waarp-r66server {hostid} loadconf
```

1.6.2 Lancer l'instance serveur

Une fois les étapes précédents effectuées vous pouvez démarer un serveur WaarpR66

```
waarp-r66server {hostid} start
```

Le chapitre suivant aborde les diférentes commandes des moniteurs WaarpR66 avec plus de détails.

1.7 Configuration Avancée

1.7.1 client.xml

Balise	Type (Default)	Description
comment	String	
identity		
hostid	String	Identifiant pour connection non SSL
sslhostid	String	Identifiant pour connection SSL
cryptokey	DesFile	Fichier de la clef DES pour le chiffrement des mots de passe
authentfile	XMLFile	Fichier d'authentification aux partenaires
ssl (Optional)		
keypath	JKSFile	JKS KeyStore pour les accès R66 via SSL (Contient le certificat serveur)
keystorepass	String	Mot de passe du JSK <keypath>
keypass	String	Mot de passe du Certificat du jks <keypath>
trustkeypath	JKSFile	JKS TrustStore pour les accès R66 via SSL en utilisant l'authentification client (contient les certificats client)
trustkeystorepass	String	Mot de passe du JSK <trustkeypath>
trustuseclientauthenticate	Boolean	Si vrai, R66 n'acceptera que les clients authorisés via SSL
directory		
serverhome	Directory	Dossier racine du moniteur R66 (Les chemins sont calculés depuis ce dossier)
in	String (IN)	Dossier par défaut de reception
out	String (OUT)	Dossier par défaut d'envoie
arch	String (ARCH)	Dossier par défaut d'archive
work	String (WORK)	Dossier par défaut de travail
conf	String (CONF)	Dossier par défaut de configuration
db		
dbdriver	String	Driver JDBC à utiliser pour se connecter à la base de données (mysql, postgresql, h2)
dbserver	String	URI JDBC de connection à la base de données (jdbc:type://{[}host:port]....). Veuillez vous référer à la documentation de votre base de donnée pour la syntaxe correcte
dbuser	String	Utilisateur à utiliser pour se connecter à la base de données
dbpasswd	String	Mot de passe de l'utilisateur
extendTaskFactory		
extendedtaskfactories	String (vide)	Liste (séparée par des virgules) des TaskFactory en tant qu'extension pour ajouter des tâches à WaarpR66

1.7.2 server.xml

Balise	Type (Default)	Description
comment	String	
identity		
hostid	String	Identifiant pour connection non SSL
sslhostid	String	Identifiant pour connection SSL
cryptokey	Des-File	Fichier de la clef DES pour le chiffrement des mots de passe
authentfile	XML-File	Fichier d'authentification aux partenaires
server		
serveradmin	String	Login pour l'accès administrateur
serverpasswd	String	Mot de passe pour l'accès administrateur chiffré par <cryptokey>
serverpasswdfile	String	Fichier PGP pour l'accès administrateur chiffré par <cryptokey> (Choisir <serverpasswd> ou <serverpasswdfile>)
usenossl	Boolean (True)	Autorise les connections non SSL
usessl	Boolean (False)	Autorise les connections SSL (voir configuration SSL plus bas)
usehttpcomp	Boolean (False)	Authorise la compression HTTP pour l'interface d'administration (HTTPS)
uselocalexec	Boolean (False)	Si vrai utilise R66 LocalExec Daemon au lieu de System.exec()
lexecaddr	Address (127.0.0.1)	Addresse du Daemon LocalExec
lexecport	Integer (9999)	Port du Daemon LocalExec
httpadmin	Directory	Dossier racine de l'interface d'administration (HTTPS)
admkeypath	JKS-File	JKS KeyStore pour les accès administrateur en HTTPS (Contient les certificats serveur)
admkeystorepass	String	Mot de passe du <admkeypath> KeyStore
admkeypass	String	Mot de passe du certificat serveur dans <admkeypath>
checkaddress	Boolean (False)	R66 vérifiera l'addresse IP distance pendant l'acceptation de la connection
checkclientaddress	Boolean (False)	R66 vérifiera l'addresse IP distance du client distant
multiplemonitors	Integer (1)	Défini le nombre de serveur du même groupe considéré comme une unique instance R66
businessfactorynetwork	String	Nom de classe complet de la Business Factory. (org.waarp.openr66.context.R66DefaultBusinessFactory)
network		
serverport	Integer (6666)	Port utilisé pour les connections en clair
serveraddresses	String (null)	Adresses utilisées pour le protocole R66 (séparées par des virgules)
serversslport	Integer (6667)	Port utilisé pour les connections chiffrées
serverssladdresses	String (null)	Adresses utilisées pour le protocole R66 en SSL (séparées par des virgules)
serverhttpport	Integer (8066)	Port de l'interface HTTP de monitoring, 0 désactive cette interface
serverhttpaddresses	String (null)	Adresses utilisées pour l'interface web de supervision (séparées par des virgules)
serverhttpsport	Integer (8067)	Port de l'interface HTTPS d'administration, 0 désactive cette interface

suite sur la page suivante

Tableau 1 – suite de la page précédente

Balise	Type (Default)	Description
serverhttpsaddresses	String (null)	Adresses utilisées pour l'interface web HTTPS d'administration (séparées par des virgules)
serverrestport	Integer (-1)	Port de l'API REST HTTP(S), -1 désactive cette interface
ssl (Optional)		
keypath	JKS-File	JKS KeyStore pour les accès R66 via SSL (Contient le certificat serveur)
keystorepass	String	Mot de passe du JSK <keypath>
keypass	String	Mot de passe du Certificat du jks <keypath>
trustkeypath	JKS-File	JKS TrustStore pour les accès R66 via SSL en utilisant l'authentification client (contient les certificats client)
trustkeystorepass	String	Mot de passe du JSK <trustkeypath>
trustuseclientauthenticate	Boolean	Si vrai, R66 n'acceptera que les clients authorisés via SSL
directory		
serverhome	Directory	Dossier racine du moniteur R66 (Les chemins sont calculés depuis ce dossier)
in	String (IN)	Dossier par défaut de reception
out	String (OUT)	Dossier par défaut d'envoie
arch	String (ARCH)	Dossier par défaut d'archive
work	String (WORK)	Dossier par défaut de travail
conf	String (CONF)	Dossier par défaut de configuration
limit		
serverthread	Integer (n*2 + 1)	Nombre de threads serveur (n=Nombre de coeur)
clientthread	Integer (10*n)	Nombre de threads client
memorylimit	Integer (1000000000)	Limite mémoire des services HTTP et REST
sessionlimit	Integer (1GB)	Limitation de bande passante par session (1GB)
globallimit	Integer (100GB)	Limitation de bande passante globale (100GB)
delaylimit	Integer (10000)	Interval entre 2 vérification de bande passante
runlimit	Integer (1000)	Limite du nombre de transfers actifs (maximum 50000)
delaycommand	Integer (5000)	Interval entre 2 execution du Commander (5s)
delayretry	Integer (30000)	Interval avant une nouvelle tentative de transfert en cas d'erreur (30s)
timeoutcon	Integer (30000)	Interval avant l'envoie d'un Time Out (30s)
blocksize	Integer (65536)	Taille des blocs (64Ko). Une valeur entre 8 ko et 16 Mo est recommandé
gaprestart	Integer (30)	Nombre de blocs doublonnés en cas d'arrêt puis reprise d'un transfert
usenio	Boolean (False)	Support NIO des fichiers. Paramètre obsolète
usecpulimit	Boolean (False)	Limitation du CPU via la gestion de la bande passante
usejdkcpulimit	Boolean (False)	Limitation CPU basé sur le JDSK natif, sinon Java Sysmon library est utilisé
cpulimit	Decimal (0.0)	% de CPU, 1.0 ne produit aucune limite
connlimit	Integer (0)	Limitation du nombre de connection
digest	Integer (2)	Utilisation d'un Digest autre que MD5 (7 pour SHA-512 recommandé)
usefastmd5	Boolean (False)	Utilisation de la bibliothèque FastMD5 (paramètre obsolète)

suite sur la page suivante

Tableau 1 – suite de la page précédente

Balise	Type (Default)	Description
fastmd5	SODLL	Path vers la JNI. Si vide, la version core de Java sera utilisée
checkversion	Boolean (True)	Utilisation du protocole etendu (>= 2.3), accès à plus de retour d'information en fin de transfert
globaldigest	Boolean (True)	Utilisation d'un digest global (MD5, SHA1, ...) par transfert de fichier
localdigest	Boolean (True)	Utilisation d'un digest local (MD5, SHA1, ...) en fin de transfert (optionnel)
compression	Boolean (False)	Active ou Désactive la compression à la volée par bloc, puis en fonction du partenaire
db		
dbdriver	address	Driver JDBC à utiliser pour se connecter à la base de données (mysql, postgresql, h2)
dbserver	String	URI JDBC de connection à la base de données (jdbc:type://{[}host:port]....). Veuillez vous référer à la documentation de votre base de donnée pour la syntaxe correcte
dbuser	String	Utilisateur de la base de données
dbpasswd	String	Mot de passe de la base de données
dbcheck	Boolean (True)	Vérification de la base de données au démarage
taskrunnernodb	Boolean (False)	WaarpR66 serveur sans base, utilise les fichiers comme information permanente sur les tâches de transfert
rest		
restssl	Boolean (False)	Utilisation de SSL par l'interface REST
restdelete	Boolean (False)	Authorisation de DELETE par l'interface REST
restauthenticated	Boolean (False)	Utilisation de l'authentification par l'interface REST
resttimelimit	Long (-1)	Time out de l'interface REST
restauthkey	Path	Clef d'authentification SHA 256 de l'interface REST
business		
businessid	String	L'hostid (1 by 1) authorisé à utiliser des Business Request
roles		
role	Array	Remplace le rôle de l'hôte en base de données
roleid	String	L'hostid (1 à 1) concerné par le remplacement
roleset	StringArray	Les nouveaux rôle attribués
aliases		
alias	Array	Permets d'utiliser des alias au lieu des hostid
realid	String	Hostid aliassé (l'alias est local)
aliasid	StringArray	L'ensemble des alias de l'hostid
extendTaskFactory		
extendedtaskfactories	String (vide)	Liste (séparée par des virgules) des TaskFactory en tant qu'extension pour ajouter des tâches à WaarpR66
pushMonitor		
Partie commune		
url	String (null)	URL de base pour les exports du moniteur en mode POST HTTP(S) JSON
delay	Integer (1000)	Délai entre deux vérifications de changement de statuts sur les transferts
intervalincluded	Boolean (True)	Si « True », les informations de l'intervalle utilisé seront fournies

suite sur la page suivante

Tableau 1 – suite de la page précédente

Balise	Type (Default)	Description
transformlongasstring	Boolean (False)	Si « True », les nombres « long » seront convertis en chaîne de caractères, sinon ils seront numériques
token	String (null)	Spécifie si nécessaire le token dans le cadre d'une authentification via Token
apiKey	String (null)	Spécifie si nécessaire le password dans le cadre d'une authentification via ApiKey (format `apiId:apiKey`)
Partie API REST		
endpoint	String (null)	End point à ajouter à l'URL de base
keepconnection	Boolean (True)	Si « True », la connexion HTTP(S) sera en Keep-Alive (pas de réouverture sauf si le serveur la ferme), sinon la connexion sera réinitialisée pour chaque appel
basicAuthent	String (null)	Spécifie si nécessaire l'authentification basique
Partie Elasticsearch		
index	String (null)	Contient le nom de l'index avec de possibles substitutions, dont `%%WARPHOST%%` pour le nom du host concerné, et les `%%DATETIME%%`, `%%DATEHOUR%%`, `%%DATE%%`, `%%YEARMONTH%%`, `%%YEAR%%` pour des substitutions de date et heure partiellement (`yyyy.MM.dd.HH.mm` à `yyyy`)
prefix	String (null)	Spécifie si nécessaire un prefix global dans le cas d'usage d'un Proxy devant Elasticsearch
username	String (null)	Spécifie si nécessaire le username (et son password) dans le cadre d'une authentification basique
paswd	String (null)	Spécifie si nécessaire le password dans le cadre d'une authentification basique
compression	Boolean (True)	Spécifie si les flux sont compressés (par défaut True)

Les balises <roles> et <aliases> contiennent des listes d'option. Exemple :

```
...
<roles>
  <role>
    <roleid>DummyHost1</roleid>
    <roleset>RoleA</roleset>
  </role>
  <role>
    <roleid>DummyHost2</roleid>
    <roleset>RoleA RoleC</roleset>
  </role>
  <role>
    <roleid>DummyHost3</roleid>
    <roleset>RoleC RoleD RoleE</roleset>
  </role>
</roles>
<aliases>
  <alias>
    <realid>DummyHost1</realid>
    <aliasid>AliasC</aliasid>
  </alias>
  <alias>
    <realid>DummyHost4</realid>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <aliasid>AliasA AliasB</aliasid>
  </alias>
</aliases>
...
```

1.7.3 Optimisation

Il peut être nécessaire de paramétrer finement dans certains cas.

Limitation de la mémoire

Il est possible de limiter l'usage de la mémoire en usant des paramètres suivants :

Limitation des services

- Services R66 : un des protocoles au moins doit être activé (TLS ou no TLS) ; si l'un des deux n'est pas utile, vous pouvez le désactiver (*usenossl* ou *usessl* à *False*)
- *uselocalexec* : à *False* si aucun usage (exécution dans un processus externes des commandes EXECxxx) (valeur par défaut)
- *serverhttpport* : si le monitoring HTTP est sans usage, vous pouvez le désactiver (*0*)
- *serverhttpsport* : si le moteur d'administration HTTPS est sans usage, vous pouvez le désactiver (*0*) (non recomandé)
- *serverrestport* : si le moteur REST est sans usage, vous pouvez le désactiver (*-1*, valeur par défaut)
- *usethrift* : si le moteur THRIFT est sans usage, vous pouvez le désactiver (*0*, valeur par défaut)
- *pushMonitor* : si le Push Monitoring Exporter est sans usage, ne pas le déclarer

Limitation des ressources

- *serverthread* : Possibilité de limiter le nombre de Threads dédiées à la partie serveur (y compris 1)
- *clientthread* : Possibilité de limiter le nombre de Threads dédiées à la partie protocolaire (il est avisé de ne par mettre moins de 10)
- *memorylimit* : Possibilité de limiter la taille mémoire maximale allouable pour décoder/encoder les pages HTTP et les réponses REST (minimum conseillé 100 Mo)
- *runlimit* : Possibilité de limiter le nombre de transferts simultanés (il est avisé de ne pas mettre moins de 2)
- *compression* : Possibilité de ne pas activer la compression à la volée (moins de mémoire et de cpu)
- de limiter l'impact processeur via une gestion adaptative de la bande passante globale :
  - *usecpulimit* à *True* : ceci active la fonctionnalité
  - *usejdkcpulimit* de préférence, laisser à *False* ou *ignoré* (permet de choisir l'implémentation sous-jacente analysant les ressources CPU)
  - *cpulimit* avec une valeur maximale autorisée pour la charge globale CPU, tous coeurs confondus (minimum conseillé *0.2*, en pratique *0.5* comme minimum) ; cette valeur détermine le seuil à partir duquel la bande passante globale sera progressivement diminuée afin de réduire l'activité CPU, puis remontée
  - *connlimit* en laissant à *0* ou *ignoré* (permet de limiter le nombre maximum de connexion mais souvent trop restrictif)

Performances

- Usage de règles dans un mode sans empreinte par paquet de données (`SENDMODE` = 1, `RECVMODE` = 2) au lieu des modes avec empreinte par paquet de données (`SENDMD5MODE` = 3, `RECVMD5MODE` = 4) (environ 15% de gains)
- *blocksize* : Possibilité d'augmenter la taille par défaut de 64KB à par exemple 256KB (en pratique, inutile d'aller au-delà), permettant de diminuer le nombre de paquets de données ainsi émis (uniquement valable sur de gros transferts)
- *gaprestart* : Possibilité de diminuer la valeur par défaut (*30*) à *10*, permettant ainsi de restreindre la réémission des paquets à la reprise du transfert (au lieu de *30 x blocksize*, ce sera par exemple *10 x blocksize*)
- *digest* : Possibilité de choisir des algorithmes plus performants (*CRC32 `=0, `MD5 `=2) ou avec moins de risques de collisions ( `SHA-XXX* tel que *SHA-512 `=7) ( `SHA-512* est conseillé car très efficace)
  - *CRC32* est le plus performant (95% avec 6ms JDK11, 10ms JDK8) mais avec le plus de collisions,
  - *MD5* performant (55% avec 88ms JDK11, 105ms JDK8) mais avec encore des collisions

    - *SHA-512* est le plus performant des SHA (au moins 25% avec 70ms JDK11, 153ms JDK8) et aux collisions infimes
    - *chiffres comparés à `SHA-256` (159ms JDK11, 192ms JDK8)*
  - *globaldigest* : Possibilité de le désactiver mais recommandé à *True* (environ 25% de gains)
  - *localdigest* : Possibilité de le désactiver (*False*) (environ 20% de gains)
  - *runlimit* : Possibilité d'augmenter ou de diminuer la valeur par défaut (1000) entre 2 et 50000 transferts concurrents
  - *compression* : Permet d'activer (désativée par défaut) la possibilité de compression à la volée des blocs et donc la vitesse des transferts sur des environnements à réseau contraint

La performance d'autres éléments peuvent jouer :

- La vitesse du processeur et de la mémoire
  - Il est conseillé de disposer d'au moins 2 coeurs et au moins 2 Go de mémoire disponible totalement pour Waarp, une valeur optimale étant 4 coeurs et 8 Go de mémoire
- La vitesse du stockage sur lequel sont écrits les fichiers (limite naturelle du transfert)
  - Il est conseillé de disposer de disques très rapides (SSD ou FC). La vitesse en lecture (émission) ou en écriture (réception) peuvent en être impactées. Ceci concerne a minima le répertoire *WORK* et *IN* et dans une moindre mesure (lecture) *OUT*.
- La vitesse et la latence du réseau sur lequel transite les données (limite naturelle du transfert)

Mini-Benchmark

Sur un Core I7 génération 5, 16 Go de mémoire, un disque rapide SSD de portable, un réseau local (*lo*), en condition complète de vérification de cohérences (*digest* à *SHA-512* (7), *globaldigest* et *localdigest* à *True*, et règle avec empreinte par paquet), les transferts ont pu atteindre 65 MB/s (520 Mbits/s).

En réduisant les vérifications de cohérence (*digest globaldigest* maintenus mais *localdigest* à *False* et règle sans empreinte par paquet), les performances sont montées à 80 MB/s (640 Mbits/s).

En supprimant toutes les vérifications de cohérence sauf celles des empreintes par paquet, le débit atteint était de 110 MB/s (880 Mbits/s) (*ceci correspond au maximum du débit disque en écriture*).

Il est fortement déconseillé de désactiver totalement toutes les vérifications de cohérence, car il ne pourra alors pas être assuré que le fichier transmis le sera sans défaut lors du transport (même si le protocole s'appuie sur TCP/IP, il est possible d'avoir une corruption sur le réseau).

Benchmarks Waarp R66

Les benchmarks suivants ont été réalisés sur un seul serveur à chaque fois, hébergeant tous les services (Waarp R66 et base de données PostgreSQL).

Modèle	TLS	NoTLS	Accélération	Description
Loop 2 coeurs	100/s	104/s	Référence	2 Serveurs en ping pong pour une taille moyenne de 250 Ko
Loop 2 coeurs	95/s	100/s	-4%	2 Serveurs en ping pong pour une taille moyenne de 250 Ko et Monitoring en mode PUSH REST
Loop 4 coeurs	127/s	133/s	27%	2 Serveurs en ping pong pour une taille moyenne de 250 Ko
Loop 4 coeurs	125/s	125/s	25%	2 Serveurs en ping pong pour une taille moyenne de 250 Ko et Monitoring en mode PUSH REST
Cluster 2 coeurs	70/s	72/s	Référence	Mode Cluster avec 1 seul serveur pour une taille moyenne de 250 Ko
Cluster 2 coeurs	83/s	87/s	21%	Mode Cluster avec 2 serveurs pour une taille moyenne de 250 Ko
Cluster 4 coeurs	89/s	91/s	27%	Mode Cluster avec 1 seul serveur pour une taille moyenne de 250 Ko
Cluster 4 coeurs	191/s	192/s	167%	Mode Cluster avec 2 serveurs pour une taille moyenne de 250 Ko
Gros Fichier 2c	152 MB/s	181 MB/s	Référence	Transfert d'un fichier de 500 Mo
Gros Fichier 4c	250 MB/s	296 MB/s	64%	Transfert d'un fichier de 500 Mo

L'évolution selon les versions depuis la 3.0 jusqu'à la dernière version.

Contexte	Nb vCore	TLS	Transferts/s	CPU	Gain
V3.0 Loop 2 Serveurs	4	Oui	30/s	100%	Référence
V3.2 Loop 2 Serveurs	4	Oui	60/s	100%	100%
V3.5.2 Loop 2 Serveurs	4	Oui	71/s	100%	137%
V3.6.0 Loop 2 Serveurs	4	Oui	100/s	90%	233%
V3.6.0 Loop 2 Serveurs Compres	4	Oui	99/s	80%	230%
V3.6.0 Loop 2 Serveurs	8	Oui	127/s	30%	323%
V3.6.0 Loop 2 Serveurs Compres	8	Oui	127/s	35%	323%
V3.6.0 Loop 2 Serveurs	4	Non	104/s	80%	Référence
V3.6.0 Loop 2 Serveurs Compres	4	Non	103/s	75%	-1%
V3.6.0 Loop 2 Serveurs	8	Non	133/s	30%	28%
V3.6.0 Loop 2 Serveurs Compres	8	Non	127/s	35%	22%
V3.6.0 Loop 2 Serveurs Monitor	4	Oui	95/s	90%	-8%
V3.6.0 Loop 2 Serveurs Monitor	8	Oui	125/s	45%	20%
V3.6.0 Loop 2 Serveurs Monitor	4	Non	100/s	85%	-4%
V3.6.0 Loop 2 Serveurs Monitor	8	Non	125/s	40%	20%
V3.6.0 Cluster 1 Serveurs	4	Oui	70/s	80%	Référence
V3.6.0 Cluster 2 Serveurs	4	Oui	83/s	100%	19% (-17% vs mono serveur)
V3.6.0 Cluster 1 Serveurs	8	Oui	89/s	30%	27%
V3.6.0 Cluster 2 Serveurs	8	Oui	191/s	60%	173% (51% vs mono serveur)
V3.6.0 Cluster 1 Serveurs	4	Non	72/s	80%	Référence
V3.6.0 Cluster 2 Serveurs	4	Non	87/s	100%	21% (-16% vs mono serveur)
V3.6.0 Cluster 1 Serveurs	8	Non	91/s	30%	26%
V3.6.0 Cluster 2 Serveurs	8	Non	192/s	60%	167% (44% vs mono serveur)

Il ressort de ces benchmarks qu'il est important d'avoir au moins 4 core (8 threads) dédiés par serveur Waarp R66 pour être optimal. En terme de mémoire, 4 GB étaient alloués à chaque instance, si possible 8 GB.

Benchmarks Waarp Gateway FTP et Waarp FTP Server

Il s'agit de benchmarks orientés FTP (Serveur ou Gateway).

Modèle	Active	Passive	Accélération	Description
FTP Natif 2 core	102/s	68/s	Référence	Petits transferts séquentiels avec reconnexion
FTP Natif 4 core	118/s	77/s	16%	Petits transferts séquentiels avec reconnexion
GW FTP 2 core	101/s	67/s	-1%	Petits transferts séquentiels avec reconnexion
GW FTP 4 core	113/s	77/s	11%	Petits transferts séquentiels avec reconnexion
GW FTP 4 core Postgre	113/s	77/s	11%	Petits transferts séquentiels avec reconnexion

Modèle	Mixte Active / Passive	Accélération	Description
FTP 10 clients 2c	488/s	Référence	10 clients avec transferts concurrents
FTP 50 clients 2c	1102/s	Référence	50 clients avec transferts concurrents
FTP 100 clients 2c	1158/s	Référence	100 clients avec transferts concurrents
FTP 10 clients 4C	1056/s	116%	10 clients avec transferts concurrents
FTP 50 clients 4C	3233/s	193%	50 clients avec transferts concurrents
FTP 100 clients 4c	4200/s	263%	100 clients avec transferts concurrents
GW FTP 10 clients 2C	234/s	Référence	10 clients avec transferts concurrents
GW FTP 50 clients 2C	260/s	Référence	50 clients avec transferts concurrents
GW FTP 100 clients 2c	244/s	Référence	100 clients avec transferts concurrents
GW FTP 10 clients 2C	224/s	-4%	10 clients avec transferts concurrents avec PostGreSQL
GW FTP 50 clients 2C	260/s	0%	50 clients avec transferts concurrents avec PostGreSQL
GW FTP 100 clients 2c	244/s	0%	100 clients avec transferts concurrents avec PostGreSQL
GW FTP 10 clients 4C	383/s	64%	10 clients avec transferts concurrents
GW FTP 50 clients 4C	1234/s	375%	50 clients avec transferts concurrents
GW FTP 100 clients 4c	1350/s	453%	100 clients avec transferts concurrents

Il ressort de ces benchmarks qu'il est important d'avoir au moins 4 core (8 threads) dédiés par serveur Waarp Gateway FTP pour être optimal. En terme de mémoire, 4 GB étaient alloués à chaque instance.

A noter que le client Waarp (basé sur FTP4J) est plus performant que l'implémentation Apache.

CHAPITRE 2

Administration

2.1 Mise à jour

2.1.1 Avec les packages

Pour une installation faite à partir des packages, utiliser une des commandes suivantes (selon la distribution) :

```
# Avec les dépôts
yum install waarp-r66

# avec le package rpm
yum install path/to/waarp-r66.rpm
```

Pour la mise à jour de la base, il est utile de lancer la commande suivante :

```
waarp-r66server {hostid} initdb -upgradeDb
```

2.1.2 Avec les archives autonomes

Pour une installation faite à partir les packages autonomes, la procédure est la suivante :

1. Si le serveur R66 ou filewatcher a été installé en tant que service, arrêter celui-ci. Pour Windows seulement : désinstaller le service.
2. Extraire l'archive au même niveau que l'ancienne installation.
3. Copier le contenu du dossier `etc` de l'ancienne installation vers le dossier `etc` de la nouvelle version.
4. Procéder de même avec le dossier data de l'ancienne installation.
5. Si le serveur R66 ou filewatcher a été installé en tant que service :
   — Pour windows seulement : réinstaller les services depuis la nouvelle installation.
   — Pour linux : mettre à jour les chemins du service avec les nouveaux dossiers.
6. Mettre à jour le schéma de la base (si elle est utilisée) :

— `waarp-r66server {hostid} initdb -upgradeDb`
Enfin, redémarrer les services.

2.2 Gestion des logs

Selon la méthode d'installation, les logs sont écrits dans le dossier `/var/log/waarp/HOSTID` ou `data/HOSTID/log`.

Les logs sont configurables dans les fichiers `/etc/waarp/HOSTID/logback-*.xml` ou `etc/donf.d/HOSTID/logback-*.xml`. Ces fichiers permettent notamment de définir l'emplacement des logs, les paramètres de rotation des logs et le niveau de verbosité.

Le détail des options de configuration est disponible sur le site de la librairie logback.

2.3 Purge de l'historique des transferts

2.3.1 En ligne de commande

Pour purger l'historique des transferts en ligne de commande, il faut executer la *commande d'export de l'historique* avec l'argument `-purge`.

Tous les arguments de la commande d'export peuvent s'appliquer pour filtrer la purge.

Par exemple, pour purger de l'historique les transferts datant d'avant le 1er juin 2018, la commande suivante peut être utilisée :

```
# Installation avec les packages
waarp-r66client HOSTID log-export -purge -stop 201806010000

# installation avec les archives portables
./bin/waarp-r66client.sh HOSTID log-export -purge -stop 201806010000
```

L'historique est eporté en XML dans le dossier `arch` de l'instance avant d'être définitivement purgé de la base de données.

2.3.2 Avec l'API REST

L'API REST de Waarp R66 expose un point d'entrée qui permet d'exporter et de purger l'historique de transfert.

Voir aussi :

— *La documentation l'API*

Par exemple, pour purger de l'historique les transferts datant d'avant le 1er juin 2018, la commande suivante peut être utilisée :

```
$ cat <<EOT | curl https://[IP SERVEUR]:8088/log -X GET -d @-
{
   "@class": "org.waarp.openr66.protocol.localhandler.packet.json.LogJsonPacket",
   "requestUserPacket": 16,
   "purge": true,
   "stop": 1527804000
}
EOT
```

Réponse du serveur pour cet exemple :


```
{
   "X-method":"GET",
   "path":"/log",
   "base":"log",
   "uri":{},
   "answer":{
      "@model":"Log",
      "results":[{
         "@class":"org.waarp.openr66.protocol.localhandler.packet.json.
↪LogResponseJsonPacket",
         "comment":null,
         "requestUserPacket":16,
         "purge":true,
         "clean":false,
         "statuspending":false,
         "statustransfer":false,
         "statusdone":false,
         "statuserror":false,
         "rule":null,
         "request":null,
         "start":null,
         "stop":1399760601400,
         "startid":null,
         "stopid":null,
         "command":16,
         "filename":"data/server1/arch/server1_1521651615878_runners.xml",
         "exported":0,
         "purged":0
         }]
      },
      "command":"GetLog",
      "message":"OK",
      "code":200
   }
```

CHAPITRE 3

Utilisation

3.1 Waarp R66 FileWatcher

3.1.1 Introduction

Le filewatcher est un mode particulier du client Waarp R66 : une fois démarré, celui-ci surveille un ou plusieurs répertoires et envoit tous les fichiers qui y sont déposés selon les données paramétrées (notamment le destinataire et la règle de transfert).

Un même filewatcher peut surveiller plusieurs répertoires ou arborescences et envoyer les fichiers à un ou plusieurs destinataires.

3.1.2 Paramétrage

Principe

Le filewatcher étant un client, il prend sa configuration dans le fichier de configuration du client `client.xml` de l'instance concernée. Selon la méthode d'installation et le système d'exploitation cible, celui-ci peut se trouver aux emplacements suivants :

— `etc/conf.d/`*`HOSTID`*`/client.xml`
— `/etc/conf.d/`*`HOSTID`*`/client.xml`
— `etc\conf.d{HOSTID}\client.xml`

Configuration minimale

Pour ajouter un filewatcher à un client Waarp R66, il suffit d'ajouter un bloc XML <spooleddaemon> à son fichier de configuration à la fin du fichier, juste avant la balise </config> :

```
<config>

  <!-- ... -->

  <spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
  </spooleddaemon>
</config>
```

Le fichier indiqué dans la balise <stopfile> permet d'arrêter le filewatcher. Une fois lancé, et tant que ce fichier n'existe pas, le filewatcher fonctionne. dès que le fichier est créé, le filewatcher s'arrête.

La configuration individuelle des dossiers à surveiller ainsi que les paramètres d'envoi des fichiers qui y sont déposés sont définis dans des blocs XML <spooled> ajoutés à <spooleddaemon>.

A minima, un bloc <spooled> doit contenir :

- un identifiant défini par l'administrateur dans une balise <name>
- un serveur r66 destinataire dans une balise <to>
- la règle de transfert à utiliser dans une balise <rule>
- un dossier à suirveiller dans une balise <directory>
- un fichier de statut dans une balise <statusfile>. Ce fichier est utilisé pour enregistrer les informations sur les fichiers déposés dans le dossier surveillé.

Par exemple :

```
<config>

  <!-- ... -->

  <spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
    <spooled>
      <name>identifiant</name>
      <to>host1</to>
      <rule>rulespooled</rule>
      <statusfile>data/HOSTID/log/status_identifiant.json</statusfile>
      <directory>data/HOSTID/spooled/out</directory>
    </spooled>
  </spooleddaemon>
</config>
```

Configuration avancée

Surveiller plusieurs dossiers

Pour surveiller plusieurs dossiers avec des paramètres d'envoi (destinataires et/ou règle de transfert) différents, il est possible de définir plusieurs blocs <spooled>. Par exemple :

```
<config>

  <!-- ... -->

  <spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
    <spooled>
      <name>identifiant</name>
      <to>host1</to>
      <rule>rulespooled</rule>
      <statusfile>data/HOSTID/log/status_identifiant.json</statusfile>
      <directory>data/HOSTID/spooled/out</directory>
    </spooled>

    <spooled>
      <name>identifiant2</name>
      <to>host2</to>
      <rule>rulespooled2</rule>
      <statusfile>data/HOSTID/log/status_identifiant2json</statusfile>
      <directory>data/HOSTID/spooled/out2/directory>
    </spooled>
  </spooleddaemon>
</config>
```

Il est également possible, pour les mêmes paramètres d'envois, de surveiller plusieurs dossiers en spécifiant plusieurs balises <directory> :

```
<config>

  <!-- ... -->

  <spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
    <spooled>
      <name>identifiant</name>
      <to>host1</to>
      <rule>rulespooled</rule>
      <statusfile>data/HOSTID/log/status_identifiant.json</statusfile>
      <directory>data/HOSTID/spooled/out</directory>
      <directory>data/HOSTID/spooled/out2/directory>
    </spooled>
  </spooleddaemon>
</config>
```

Envoi des fichiers à plusieurs destinataires

Pour envoyer les fichiers déposés dans un dossier surveillé à plusieurs destinataires, il est possible de spécifier plusieurs balises <to> :

```
<config>

  <!-- ... -->

  <spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
    <spooled>
      <name>identifiant</name>
      <to>host1</to>
      <to>host2</to>
      <rule>rulespooled</rule>
      <statusfile>data/HOSTID/log/status_identifiant.json</statusfile>
      <directory>data/HOSTID/spooled/out</directory>
    </spooled>
  </spooleddaemon>
</config>
```

Autres directives de configuration

Dans les blocs XML <spooled>, il est également possible d'utiliser les balises suivantes :

<info> Métadonnées envoyées avec le fichier durant le transfert (corresponfd à l'argument `-info` de la commande d'envoi.

<regex> Une espression régulière de filtrage des fichiers à prendre en compte (permet d'exclure des fichiers des transferts).

Modifié dans la version 3.1.0 : l'expresion regulière permet de filtrer le chemin complet du fichier et non plus le nom du fichier seulement

<recursive> Si récursif, les sous dossiers aussi sont surveillés.

<elapse> L'intervalle entre 2 scan du dossier (en ms).

<submit> Si submit est True, les transferts sont asynchrones. sinon, ils sont directs.

<parallel> Si submit est false, c'est-à-dire si les transferts sont gérés directement par le file watcher, les transferts peuvent être faits en parallèle ou séquentiellement.

<limitParallel> Si les transferts doivent être faits en parallèle, le nombre maximal de transferts simultanés.

<waarp> L'historique de transfert peut être envoyé au serveur Waarp R66 désigné ci-dessous. ceci permet la consultation de l'historique du filewatcher dans l'interface HTTP de monitoring de ce serveur.

Ceci est facultatif (et inutile si submit vaut true – les transferts sont déjà effectués par un serveur – ou si les interfaces de monitoring sont désactivées).

<elapseWaarp> Si l'historique doit être envoyé à un serveur Waarp R66, intervalle en ms entre deux envois.

<ignoreAlreadyUsed> Si positionné à vrai, tout fichier déjà traité et non effacé, même s'il est modifié, sera ignoré et ne sera donc pas renvoyé pour éviter tout risque de collisions quant au contenu transféré. Normalement, cette option devrait être activée car la modification d'un fichier non transféré est une erreur d'exploitation.

Cependant, si cette option n'est pas activée ou absente, alors, même si le fichier a été pris en compte pour un transfert mais toujours non effectué (partenaire injoignable par exemple), alors le nouveau contenu prendra le dessus sur le précédent et relancera la procédure de trasfert.

Par défaut cette option est désactivée car elle ne gène pas l'usage normal.

Exemple complet

```
<spooleddaemon>
    <stopfile>data/HOSTID/log/filewatcher.stop</stopfile>
    <spooled>
        <name>identifiant</name>
        <to>host1</to>
        <to>host2</to>
        <rule>rulespooled</rule>
        <statusfile>data/HOSTID/log/status_identifiant.json</statusfile>
        <directory>data/HOSTID/spooled/out</directory>
        <directory>data/HOSTID/spooled/out2</directory>
        <regex>.*\.?ar$</regex>
        <recursive>True</recursive>
        <elapse>1000</elapse>
        <submit>False</submit>
        <parallel>True</parallel>
        <limitParallel>0</limitParallel>
        <info>spooled transfer</info>
        <waarp>hostas</waarp>
        <elapseWaarp>5000</elapseWaarp>
        <ignoreAlreadyUsed>False</ignoreAlreadyUsed>
    </spooled>
</spooleddaemon>
```

3.1.3 Lancement

Linux

Avec les archives, le filewatcher peut être démarré avec la commande suivante :

```
./bin/waarp-r66client HOSTID watcher start
```

Cette commande démarre le filelwatcher au premier plan. On peut l'arrêter en tapant `Control-C`.

Pour le démarrer en tant tâche de fond (service), il faut définir la variable `WAARP_SERVICE` avant :

```
export WAARP_SERVICE=1
./bin/waarp-r66client.sh HOSTID watcher start
```

Le service peut alors être arrêté ou redémarré avec les commandes suivantes :

```
./bin/waarp-r66client.sh HOSTID watcher stop
./bin/waarp-r66client.sh HOSTID watcher restart
```

Windows

Sous Windows, le filewatcher peut être démarré avec la commande suivante :

```
bin\waarp-r66client.bat HOSTID watcher start
```

Cette commande démarre le filelwatcher au premier plan. On peut l'arrêter en tapant `Control-C`.

Il est également possible de l'installé en tant que service système avec la commande :

```
bin\waarp-r66client.bat HOSTID install
```

On peut ensuite le démarrer, l'arrêter ou le redémarrer avec le gestionnaire de services système.

3.2 Configuration de la tâche ICAP

Nouveau dans la version 3.4.0.

Voir aussi :

La documentation de la tâche `ICAP` est disponible *ici*

3.2.1 Mode opératoire préconisé pour l'installation et la configuration

La commande `icaptest` est disponible en tant que sous commande du script *Waarp R66 Client* afin de valider le bon fonctionnement avec le serveur ICAP cible.

Prérequis :

- L'adresse du serveur ICAP et son port (noté hosticap et porticap) ; par défaut le port est 1344 et peut être omis
- Connaître le nom du service de scan associé à ce serveur (usuellement un parmi : `avscan`, `virus_scan`, `srv-clamav`). S'il est connu, il est possible d'utiliser un modèle pré-enregistré (`DEFAULT_MODEL` ou `ICAP_AVSCAN` pour `avscan`, `ICAP_CLAMAV` pour `srv-clamav`, `ICAP_VIRUS_SCAN` pour `virus_scan`).
- D'autres options spécifiques peuvent être nécessaires et seront établies selon les tests à réaliser.

Etape 1 : Permettre la connexion et l'envoi d'un fichier sain (nommé fichier)

La commande sera de la forme :

```
./bin/waarp-r66client.sh HOSTID -file fichier \
      -to hosticap [-port porticap] \
      (-service nom | -model nom) -logger DEBUG
```

Cette commande va réaliser une requête OPTIONS puis `RESPMOD` avec le serveur pour le service nommé (ou selon le modèle choisi) et afficher toutes les étapes en mode `DEBUG`.

Si le retour est 0 (sans erreur), il n'y a a priori pas d'autres options à positionner pour un cas sain.

Si le retour est en erreur, il faut analyser le code erreur et le log produit.

- code 1 : Bad arguments ; Veuillez vérifier vos paramètres car il s'agit d'une erreur dans la ligne de commande
- code 2 : ICAP protocol error ; Le serveur a mal répondu aux requêtes (le protocole ICAP est sans doute a adapter côté client, si possible)
- code 3 : Network error ; Il s'agit d'un problème purement réseau (un parefeu peut bloquer le flux par exemple)

- code 4 : Scan KO ; Soit le serveur ou le service est mal configuré, soit il faut adapter le client au code retourné par le serveur
- code 5 : Scan KO but post action required in error ; En l'état, sauf si vous avez spécifié une des options concernées (`-errorMove path` | `-errorDelete` | `-sendOnError`), cette erreur ne devrait jamais apparaître à cette étape du test

Dans les cas 2 et 4, il faut analyser ce que reçoit le client du serveur ICAP et le cas échéant adapter les modalités de traitement du client en fonction de ces retours.

Cela peut concerner les tailles limites :

- [`-previewSize size`, défaut aucun] spécifie la taille de Preview à utiliser (défaut négociée)
- [`-blockSize size`, défaut 8192] spécifie la taille en émission à utiliser (défaut 8192)
- [`-receiveSize size`, défaut 65536] spécifie la taille en réception à utiliser (défaut 65536)
- [`-maxSize size`, défaut MAX_INTEGER] spécifie la taille Max d'un fichier à utiliser (défaut MAX_INTEGER)

Cela peut concerner les valeurs de statut et les retours associés :

- [`-keyPreview key -stringPreview string`, défaut aucun] spécifie la clef et la chaîne associée pour Options à valider (défaut aucun)
- [`-key204 key -string204 string`, défaut aucun] spécifie la clef et la chaîne associée pour 204 ICAP à valider (défaut aucun)
- [`-key200 key -string200 string`, défaut aucun] spécifie la clef et la chaîne associée pour 200 ICAP à valider (défaut aucun)
- [`-stringHttp string`, défaut aucun] spécifie la chaîne pour HTTP 200 ICAP à valider (défaut aucun)

Pour rappel, la norme ICAP indique :

- OPTIONS doit retourner un code `200` et `Preview` dans les balises :
  - si le Preview est indisponible, vous pouvez forcer sa taille avec l'option `-previewSize` ce qui évitera les requêtes OPTIONS pour la suite
  - si le code est `404`, le service n'est pas connu. Il vous faut récupérer ce nom de service auprès de vos services informatiques (ICAP ne permet pas de lister les services disponibles).
- RESPMOD doit retourner :
  - un code `204` si le fichier est validé
  - un code `200` si le fichier est invalide
    - Si le fichier est sain mais qu'un code 200 est retourné, il est possible d'analyser les clefs/valeurs ICAP (`-key200`/`-string200`) ou le contenu de la partie HTTP de la réponse (`-stringHttp`)

Etape 2 : Permettre la connexion et l'envoi d'un fichier malsain de test

Si vous ne disposez pas d'un tel fichier, vous pouvez spécifier un test interne basé sur EICAR en précisant l'option `-file EICARTEST`.

```
./bin/waarp-r66client.sh HOSTID -file fichier EICARTEST \
        -to hosticap [-port porticap] \
       (-service nom | -model nom) -logger DEBUG
```

Veillez à conserver les options que vous avez introduites dans l'étape 1, y compris celles correctives.

Normalement, le retour devrait être de la valeur 4 (Scan KO).

Si tel n'est pas le cas, il faut à nouveau analyser les logs et le retour.

Par exemple, si le retour est 204, il est possible qu'une clef ICAP dise autrement (`-key204`/`-string204`).

Si le code était déjà 200 pour un fichier sain, il est possible qu'il faille modifier les options précédemment mises en place pour fonctionner tant avec un fichier sain qu'un fichier malsain (`-key200`/`-string200` et/ou `-stringHttp`).

Etape 3 : Configurer une règle pour utiliser ces paramètres

Il s'agit maintenant de créer ou modifier une règle pour y ajouter une tâche ICAP avec les paramètres valides pour votre configuration.

Par exemple :

```
<task>
  <type>ICAP</type>
  <path>-file #TRUEFULLPATH# -to hostname -model ICAP_AVSCAN
  -sendOnError -ignoreNetworkError -- -file #TRUEFULLPATH# -to
  requestedHost -rule rule -copyinfo -info FILE INFECTED</path>
  <delay>10000</delay>
</task>
```

Puis de tester, en mode DEBUG, l'exécution de cette règle suite à un transfert l'utilisant.

Voir aussi :

— Documentation de la *tâche ICAP*

Options spécifiques

Ces options sont plus spécifiques au traitement comme tâche dans R66. Elles permettent de gérer les cas d'erreurs, en assurant ce que devient le fichier (déplacé, effacer ou renvoyer vers un autre serveur) ou en ignorant des comportements réseaux instables (sur une erreur réseau) ou en ignorant les trop gros fichiers.

— [`-errorMove path` | `-errorDelete` | `-sendOnError`] spécifie l'action en cas de scan erronné : un répertoire de quarantaine, l'effacement du fichier, la retransmission (R66) vers un autre partenaire (mutuellement exclusif) (défaut aucun)
— [`-ignoreNetworkError`] spécifie que sur une erreur réseau, le fichier sera considéré comme OK
— [`-ignoreTooBigFileError`] spécifie que sur une erreur de fichier trop grand, le fichier sera considéré comme OK

3.3 Configuration du Monitoring en mode PUSH HTTP(S) REST ou vers un service Elasticsearch

Nouveau dans la version 3.6.0.

3.3.1 Description générale

Le moniteur en mode export REST JSON des états des transferts permet d'envoyer en mode POST vers un service REST HTTP(S) de son choix l'état des transferts à intervalles réguliers.

Il permet au choix de le faire vers une simple API REST ou vers un serveur Elasticsearch. La configuration est différente mais le fonctionnement est globalement le même.

Seuls les transferts ayant subi un changement sont envoyés.

Le format du JSON est comme suit :

```
{
  "results": [                          # Array of Transfer information
    {
```

(suite sur la page suivante)

(suite de la page précédente)

```
      "specialId": 12345,                    # Id as Long (-2^63 to 2^63 - 1)
      "uniqueId": "owner.requester.requested.specialId", # Unique global Id
      "hostId": "R66Owner",                  # R66 Owner (Server name)
      "globalStep": "step",                  # Global Current Step
      "globalLastStep": "laststep",          # Global Last Step previous Current
      "step": 1,                             # Current Step in Global Current Step
      "rank": 123,                           # Current Rank in transfer step
      "status": "status",                    # Current status
      "stepStatus": "stepstatus",            # Status of previous Step
      "originalFilename": "originalFilename", # Original Filename
      "originalSize": 123456,                # Original file size
      "filename": "filename",                # Resolved local filename
      "ruleName": "ruleName",                # Rule name
      "blockSize": 123,                      # Block size during transfer
      "fileInfo": "fileInfo",                # File information, containing associated␣
↪file transfer information
      "followId": 123456,                    # Follow Id as Long (-2^63 to 2^63 - 1)
      "transferInfo": "transferInfo as Json", # Transfer internal information as Json␣
↪String
      "start": "2021-03-28T11:55:15Z",       # Start date time of the transfer operation
      "stop": "2021-03-28T11:58:32Z",        # Current last date time event of the␣
↪transfer operation
      "requested": "requested",              # Requested R66 hostname
      "requester": "requester",              # Requester R66 hostname
      "retrieve": true,                      # True if the request is a Pull, False if␣
↪it is a Push
      "errorCode": "errorCode",              # Code of error as one char
      "errorMessage": "errorMessage",        # String message of current Error
      "waarpMonitor": {                      # Extra information for indexing if␣
↪necessary
        "from": "2021-03-28T11:58:15Z",      # filter from (could be empty if none)
        "to": "2021-03-28T11:59:15Z",        # filter to
        "index": "r66owner"                  # R66 Hostname lowercase
      }
    },
    ...
  ]
}
```

3.3.2 Configuration

Une seule configuration est possible. Si `index` est positionné, il s'agit d'une configuration pour Elasticsearch, sinon pour API REST.

Pour un POST sur une API REST :

```
<pushMonitor>
  <url>http://127.0.0.1:8999</url>
  <endpoint>/log</endpoint>
  <delay>1000</delay>
  <basicAuthent>basicAuthent</basicAuthent>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  <token>token</token>
  <apiKey>apiKey</apiKey>
  <keepconnection>True</keepconnection>
  <intervalincluded>True</intervalincluded>
  <transformlongasstring>False</transformlongasstring>
</pushMonitor>
```

Description des paramètres :

- `url` indique l'URL de base du service REST HTTP(S) distant
- `endpoint` indique l'extension URI du service REST HTTP(S) distant
  - Ainsi, pour l'exemple, l'URI complète sera `http://127.0.0.1:8999/log`
  - Si HTTPS est utilisé, le KeyStore et TrustStore par défaut de Waarp seront utilisés
- `delay` indique le délai en ms entre deux vérifications pour récupérer les transferts dont l'information aurait changée. Par défaut, la valeur est de `1000` ms. La valeur minimale est de 500ms.
- `keepconnection` Si « True », la connexion HTTP(S) sera en Keep-Alive (pas de réouverture sauf si le serveur la ferme), sinon la connexion sera réinitialisée pour chaque appel (défaut : False)
  - Avec la valeur `True`, les performances sont améliorées en évitant les reconnexions.
- `intervalincluded` indique si les informations de l'intervalle utilisé seront fournies (défaut : True)
- `transformlongasstring` indique si les nombres « long » seront convertis en chaîne de caractères, sinon ils seront numériques (certaines API REST ne supportent pas des long sur 64 bits) (défaut : True)
  - Utile notamment avec ELK car les nombres longs (identifiant unique) sont trop long lors du parsing et sont tronqués.
- Si une authentification est nécessaire, plusieurs options sont possibles (unique) :
  - Authentification Basic : `basicAuthent` contient l'authentification Basic au format Base 64
  - Bearer Token : `token` contenant le token d'accès
  - ApiKey : `apiKey` contenant la clef d'API sous la forme `apiId:apiKey`

Pour une indexation par Bulk sur Elasticsearch :

```
<pushMonitor>
  <url>http://127.0.0.1:8999</url>
  <prefix>/pathPrefix</prefix>
  <delay>1000</delay>
  <index>indexName</index>
  <username>username</username><paswd>password</passwd>
  <token>token</token>
  <apiKey>apiKey</apiKey>
  <intervalincluded>True</intervalincluded>
  <transformlongasstring>False</transformlongasstring>
  <compression>True</compression>
</pushMonitor>
```

Description des paramètres :

- `url` indique l'URL de base du service REST HTTP(S) distant ; plusieurs url sont possibles, séparées par ","
- `prefix` indique un prefix à ajouter à chaque requête, notamment si Elasticsearch est derrière un Proxy (non obligatoire)
- `delay` indique le délai en ms entre deux vérifications pour récupérer les transferts dont l'information aurait changée. Par défaut, la valeur est de `1000` ms. La valeur minimale est de 500ms.
- `intervalincluded` indique si les informations de l'intervalle utilisé seront fournies (défaut : True)
- `transformlongasstring` indique si les nombres « long » seront convertis en chaîne de caractères, sinon ils seront numériques (certaines API REST ne supportent pas des long sur 64 bits) (défaut : False)
  - Utile notamment avec ELK car les nombres longs (identifiant unique) sont trop long lors du parsing et sont tronqués.
- `index` contient le nom de l'index. Des substitutions sont possibles pour avoir de multiples index :

  - %%WAARPHOST%% remplacé par le nom du serveur R66
  - %%DATETIME%% remplacé par la date au format YYYY.MM.dd.HH.mm
  - %%DATEHOUR%% remplacé par la date au format YYYY.MM.dd.HH
  - %%DATE%% remplacé par la date au format YYYY.MM.dd
  - %%YEARMONTH%% remplacé par la date au format YYYY.MM
  - %%YEAR%% remplacé par la date au format YYYY
  - La date considérée est la date lors du dernier déclenchement du monitoring
  - Le nom de l'index sera en minuscule, quelque soit la casse d'origine (exigence Elasticsearch)
  - Ainsi waarpR66-%%WAARPHOST%%-%%DATE%% donnerait waarpr66-hosta-2021-06-21
- Si une authentification est nécessaire, plusieurs options sont possibles (unique) :
  - Authentification Basic : username et paswd contienent l'authentification Basic
  - Bearer Token : token contenant le token d'accès
  - ApiKey : apiKey contenant la clef d'API sous la forme apiId:apiKey
- compression spécifie si les transferts d'information vers Elasticsearch utiliseront la compression (True) ou pas (False) (défaut : True)

Dernière date de vérification

A chaque transfert réussi, le moniteur met à jour la date de référence pour la prochaine vérification dans la base dans le champ others de la configuration du Host. Ceci permet, en cas d'arrêt du serveur, d'enregistrer le dernier état et ainsi de limiter le nombre de possibles doublons qui seraient renvoyés lors du redémarrage.

Si besoin, vous pouvez modifier cette valeur directement dans la base pour refléter le timestamp à utiliser comme point de départ (lastMonitoringDateTime).

Cas particulier des clusters

Afin de ne pas publier plusieurs fois les mêmes logs, il est recommandé de n'activer cette option que sur un seul des membres du cluster.

Si celui-ci devait s'arrêter, la reprise à son redémarrage reprendra là où il en était.

Si c'est un problème plus grave (le serveur physique est indisponible), vous pouvez alors activer cette fonction en la basculant sur un autre membre du cluster.

Pour Elasticsearch

Si la connection est directe avec Elasticsearch (ou au travers d'un proxy), il n'est pas besoin d'utiliser l'option transformlongasstring, en la laissant à False par défaut.

Exemple de configuration d'un Logstash

Il est possible par exemple de router vers un service Logstash les logs JSON ainsi produits via une API REST (et non directement dans Elasticsearch).

La configuration du Logstash peut être la suivante : (avec le mode transformlongasstring as True)

```
# Waarp R66 -> Logstash -> Elasticsearch pipeline.
input {
  http {
    # default: 0.0.0.0
    host => "0.0.0.0"
    ssl => false
```

(suite sur la page suivante)

(suite de la page précédente)

```
    # default: 8080
    port => 5044
    type => "r66json"
  }
}

filter {
  if [type] == "r66json" {
    # Split from array resuts
    if !("splitted" in [tags]) {
      split {
         field => "results"
         add_tag => ["splitted"]
      }
    }
    if ("splitted" in [tags]) {
      # Move to root
      ruby {
        code => "
            event.get('results').each {|k, v|
                event.set(k, v)
            }
            event.remove('results')
        "
      }
      # Discover extra Json field
      # Change Date String as DateTime
      date {
        match => [ "start", "ISO8601" ]
        target => "start"
      }
      date {
        match => [ "stop", "ISO8601" ]
        target => "stop"
      }
      date {
        match => [ "[waarpMonitor][from]", "ISO8601" ]
        target => "[waarpMonitor][from]"
      }
      date {
        match => [ "[waarpMonitor][to]", "ISO8601" ]
        target => "[waarpMonitor][to]"
      }
      # Create index name : %{[logInfo][level]}
      mutate {
        add_field => { "[@metadata][target_index]" => "waarpr66-%{[waarpMonitor][index]}-
↪%{+YYYY.MM.dd}" }
      }
      # Remove headers from HTTP request and extra fields
      mutate {
        remove_field => [ "headers", "host", "sort", "tags", "@version" ]
      }
```

(suite sur la page suivante)

(suite de la page précédente)

```
    }
  }
}

output {
  if "r66json" in [type] {
    elasticsearch {
      hosts => ["http://127.0.0.1:9200"]
      index => "%{[@metadata][target_index]}"
      document_id => "%{uniqueId}"
      doc_as_upsert => true
      #user => "elastic"
      #password => "changeme"
    }
  }
  # Debug mode file and output
#  file {
#    path => "/tmp/logstash-R66.log"
#  }
#  stdout{
#    codec => rubydebug
#  }
}
```

On Elastic, the mapping shall be defined to ensure correct type :

- waarpMonitor.to Date
- waarpMonitor.from Date
- stop Date
- start Date
- specialId string
- followId string
- originalSize string
- hostId string
- waarpMonitor.index string
- blockSize int
- errorMessage string
- filename string
- type string
- stepStatus string
- transferInfo string
- originalFilename string
- requester string
- globalStep string
- ruleName string
- requested string
- fileInfo string
- status string
- errorCode int
- retrieve boolean
- globalLastStep string
- step int
- rank long
- uniqueId string

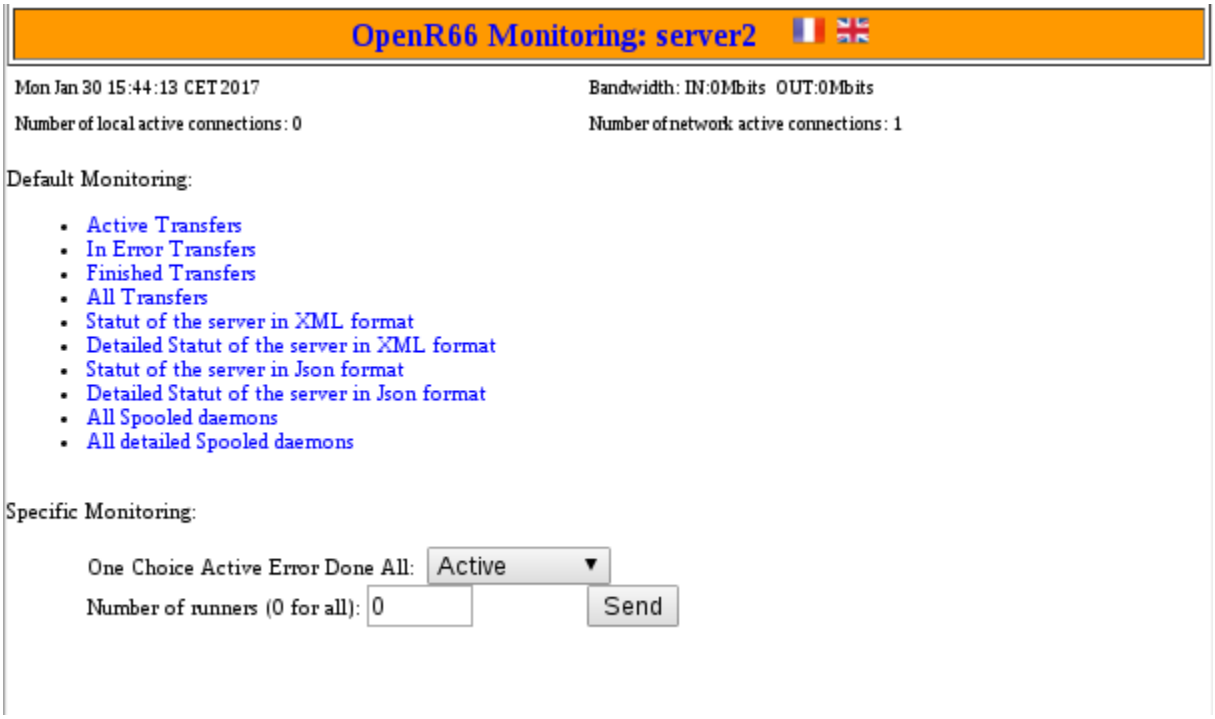
CHAPITRE 4

Interface Web

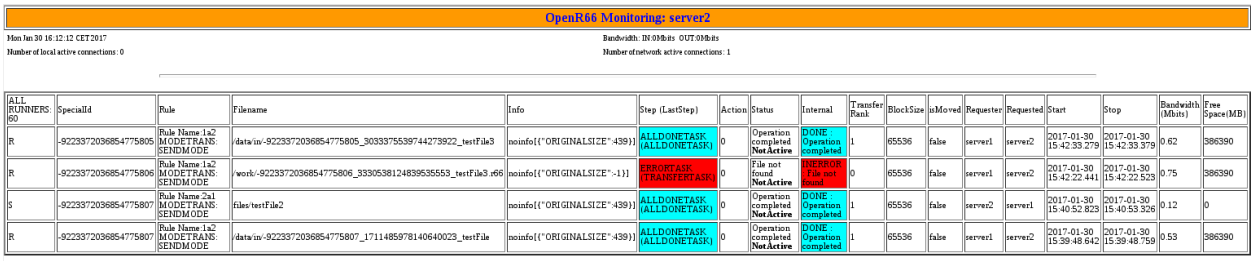
4.1 Monitoring

L'interface de monitoring permet de visualiser l'ensemble des transferts passés et en cours.

4.1.1 Accueil



4.1.2 Transferts passés



OpenR66 Monitoring: server2

Mon Jan 30 16:12:12 CET 2017 — Bandwidth: IN:0Mbits OUT:0Mbits

Number of local active connections : 0 — Number of network active connections : 1

ALL RUNNERS: 60	SpecialId	Rule	Filename	Info	Step (LastStep)	Action	Status	Internal	Transfer Rank	BlockSize	isMoved	Requester	Requested	Start	Stop	Bandwidth (Mbits)	Free Space(MB)
R	-9223372036854775805	Rule Name:1a2 MODETRANS: SENDMODE	/data/in/-9223372036854775805_3033375539744273922_testFile3	noinfo[{"ORIGINALSIZE":439}]	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE Operation completed	1	65536	false	server1	server2	2017-01-30 15:42:33.279	2017-01-30 15:42:33.379	0.62	386390
R	-9223372036854775806	Rule Name:1a2 MODETRANS: SENDMODE	/work/-9223372036854775806_3330538124839535553_testFile3.r66	noinfo[{"ORIGINALSIZE":-1}]	ERRORTASK (TRANSFERTASK)	0	File not found **NotActive**	ENERROR File not found	0	65536	false	server1	server2	2017-01-30 15:42:22.441	2017-01-30 15:42:22.523	0.75	386390
S	-9223372036854775807	Rule Name:2a1 MODETRANS: SENDMODE	files/testFile2	noinfo[{"ORIGINALSIZE":439}]	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE Operation completed	1	65536	false	server2	server1	2017-01-30 15:40:52.823	2017-01-30 15:40:53.326	0.12	0
R	-9223372036854775807	Rule Name:1a2 MODETRANS: SENDMODE	/data/in/-9223372036854775807_1711485978140640029_testFile	noinfo[{"ORIGINALSIZE":439}]	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE Operation completed	1	65536	false	server1	server2	2017-01-30 15:39:48.642	2017-01-30 15:39:48.759	0.53	386390

4.1.3 Statut serveur

Elle permet aussi d'avoir un accès au statut du server au format *XML* ou *JSON*.

Le statut simple donne accès au informations suivantes :

```
<STATUS>
  <HostID>server2</HostID>
  <Date>2017-01-30T15:53:22.782+01:00</Date>
  <LastRun>2017-01-30T15:53:22.774+01:00</LastRun>
  <FromDate>2017-01-29T15:53:22.774+01:00</FromDate>
  <SecondsRunning>869</SecondsRunning>
  <NetworkConnections>1</NetworkConnections>
  <NbThreads>114</NbThreads>
  <InBandwidth>0</InBandwidth>
  <OutBandwidth>0</OutBandwidth>
  <OVERALL>
    <AllTransfer>4</AllTransfer>
    <Unknown>0</Unknown>
    <NotUpdated>0</NotUpdated>
    <Interrupted>0</Interrupted>
    <ToSubmit>0</ToSubmit>
    <Error>1</Error>
    <Running>0</Running>
    <Done>3</Done>
    <InRunning>0</InRunning>
    <OutRunning>0</OutRunning>
    <LastInRunning>2017-01-30T15:42:33.302+01:00</LastInRunning>
    <LastOutRunning>2017-01-30T15:38:53.121+01:00</LastOutRunning>
    <InAll>3</InAll>
    <OutAll>1</OutAll>
    <InError>1</InError>
    <OutError>0</OutError>
  </OVERALL>
  <STEPS>
    <Notask>0</Notask>
    <Pretask>0</Pretask>
    <Transfer>0</Transfer>
    <Posttask>0</Posttask>
    <AllDone>3</AllDone>
    <Error>1</Error>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  </STEPS>
  <RUNNINGSTEPS>
    <AllRunning>0</AllRunning>
  </RUNNINGSTEPS>
</STATUS>
```

Le statut détaillé renvoie en plus le détails des etapes de lancement de transfert ainsi que les différentes erreurs de transfert rencontrées.

```
<STATUS>
  <HostID>server2</HostID>
  <Date>2017-01-30T15:53:10.845+01:00</Date>
  <LastRun>2017-01-30T15:53:10.834+01:00</LastRun>
  <FromDate>2017-01-29T15:53:10.834+01:00</FromDate>
  <SecondsRunning>857</SecondsRunning>
  <NetworkConnections>1</NetworkConnections>
  <NbThreads>114</NbThreads>
  <InBandwidth>0</InBandwidth>
  <OutBandwidth>0</OutBandwidth>
  <OVERALL>
    <AllTransfer>4</AllTransfer>
    <Unknown>0</Unknown>
    <NotUpdated>0</NotUpdated>
    <Interrupted>0</Interrupted>
    <ToSubmit>0</ToSubmit>
    <Error>1</Error>
    <Running>0</Running>
    <Done>3</Done>
    <InRunning>0</InRunning>
    <OutRunning>0</OutRunning>
    <LastInRunning>2017-01-30T15:42:33.302+01:00</LastInRunning>
    <LastOutRunning>2017-01-30T15:38:53.121+01:00</LastOutRunning>
    <InAll>3</InAll>
    <OutAll>1</OutAll>
    <InError>1</InError>
    <OutError>0</OutError>
  </OVERALL>
  <STEPS>
    <Notask>0</Notask>
    <Pretask>0</Pretask>
    <Transfer>0</Transfer>
    <Posttask>0</Posttask>
    <AllDone>3</AllDone>
    <Error>1</Error>
  </STEPS>
  <RUNNINGSTEPS>
    <AllRunning>0</AllRunning>
    <Running>0</Running>
    <InitOk>0</InitOk>
    <PreProcessingOk>0</PreProcessingOk>
    <TransferOk>0</TransferOk>
    <PostProcessingOk>0</PostProcessingOk>
```

(suite sur la page suivante)

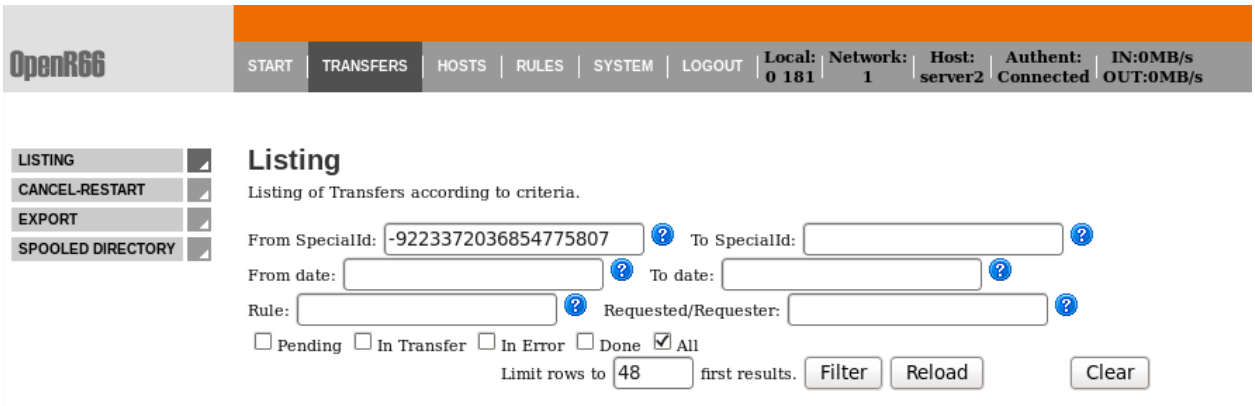
(suite de la page précédente)

```
    <CompleteOk>0</CompleteOk>
  </RUNNINGSTEPS>
  <ERRORTYPES>
    <ConnectionImpossible>0</ConnectionImpossible>
    <ServerOverloaded>0</ServerOverloaded>
    <BadAuthent>0</BadAuthent>
    <ExternalOp>0</ExternalOp>
    <TransferError>0</TransferError>
    <MD5Error>0</MD5Error>
    <Disconnection>0</Disconnection>
    <FinalOp>0</FinalOp>
    <Unimplemented>0</Unimplemented>
    <Internal>0</Internal>
    <Warning>0</Warning>
    <QueryAlreadyFinished>0</QueryAlreadyFinished>
    <QueryStillRunning>0</QueryStillRunning>
    <KnownHost>0</KnownHost>
    <RemotelyUnknown>0</RemotelyUnknown>
    <CommandNotFound>0</CommandNotFound>
    <PassThroughMode>0</PassThroughMode>
    <RemoteShutdown>0</RemoteShutdown>
    <Shutdown>0</Shutdown>
    <RemoteError>0</RemoteError>
    <Stopped>0</Stopped>
    <Canceled>0</Canceled>
    <FileNotFound>1</FileNotFound>
    <Unknown>0</Unknown>
  </ERRORTYPES>
</STATUS>
```

4.2 Administration

L'interface d'administration pertmet de controller le moniteur WaarpR66.

4.2.1 Transfert



Lister les transferts

SpecialId	Rule	Filename	Info	TransferInfo	Step (LastStep)	Action	Status	Internal	Block	isSender	isMoved	Requester	Requested	Start	Stop	Bandwidth	Free Space
-9223372036854775798	Rule Name:CONFIG_DIFF_SERVER2_2 MODETRANS: SENDMODE	/in/-9223372036854775798_3391523674041728069_server2-20161129180310.zip	noinfo	{"ORIGINALSIZE":1405}	POSTTASK (POSTTASK)	0	Current step in running **NotActive**	DONE : Transfer step ok	1	false	false	server1-ssl	server2-ssl	2016-11-29 18:03:23.62	2017-02-02 17:37:22.924	0.00	14686
-9223372036854775801	Rule Name:CONFIG_DIFF_GWFTP_3 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/gwftp-20161129180310.zip	noinfo	{"ORIGINALSIZE":1542}	ERRORTASK (NOTASK)	0	Connection Impossible **NotActive**	INERROR : Connection Impossible	0	true	false	server2-ssl	gwftp-ssl	2016-11-29 18:03:21.76	2016-11-29 18:03:56.211	0.00	0
-9223372036854775800	Rule Name:CONFIG_DIFF_GWFTP_2 MODETRANS: SENDMODE	/in/-9223372036854775800_4831282629661183737_gwftp-20161129180310.zip	noinfo	{"ORIGINALSIZE":1542}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	1	false	false	server1-ssl	server2-ssl	2016-11-29 18:03:21.526	2016-11-29 18:03:21.816	0.21	14686
-9223372036854775802	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/aaa	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	true	false	server2-ssl	gwftp-ssl	2016-02-19 15:07:46.394	2016-02-19 15:07:49.77	0.30	0
-9223372036854775802	Rule Name:toFTP MODETRANS: SENDMODE	/in/-9223372036854775802_5678557295443102166_aaa	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	false	false	server1	server2	2016-02-19 15:07:46.147	2016-02-19 15:07:46.445	3.34	14686
-9223372036854775803	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/test1.file	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	true	false	server2-ssl	gwftp-ssl	2016-02-19 15:07:12.986	2016-02-19 15:07:15.35	0.42	0
-9223372036854775803	Rule Name:toFTP MODETRANS: SENDMODE	/in/-9223372036854775803_3520374949483497526_test1.file	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	false	false	server1	server2	2016-02-19 15:07:12.618	2016-02-19 15:07:13.023	2.46	14686
-9223372036854775804	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/test10.file	noinfo	{"ORIGINALSIZE":10485760}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	160	true	false	server2-ssl	gwftp-ssl	2016-02-19 14:17:40.056	2016-02-19 14:17:46.412	1.57	0
-9223372036854775804	Rule Name:toFTP MODETRANS: SENDMODE	/in/-9223372036854775804_6817776177774190376_test10.file	noinfo	{"ORIGINALSIZE":10485760}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	160	false	false	server1	server2	2016-02-19 14:17:38.466	2016-02-19 14:17:40.08	6.19	14686
-9223372036854775805	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/test1.file	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	true	false	server2-ssl	gwftp-ssl	2016-02-19 13:55:20.365	2016-02-19 13:55:24.403	0.25	0

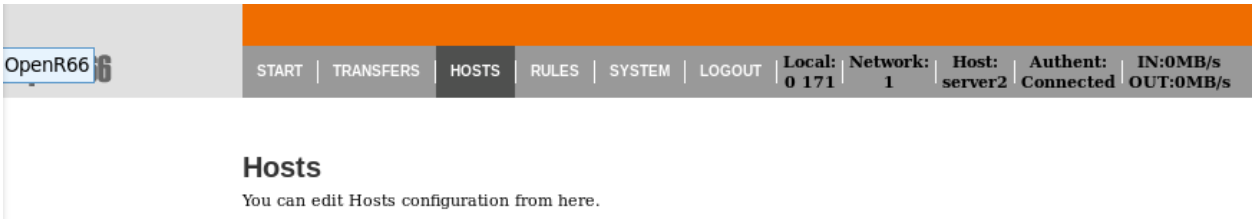
Controller les transferts

Depuis l'interface d'administration il est possible d'arrêter, annuler, ou redémarer un transferts.

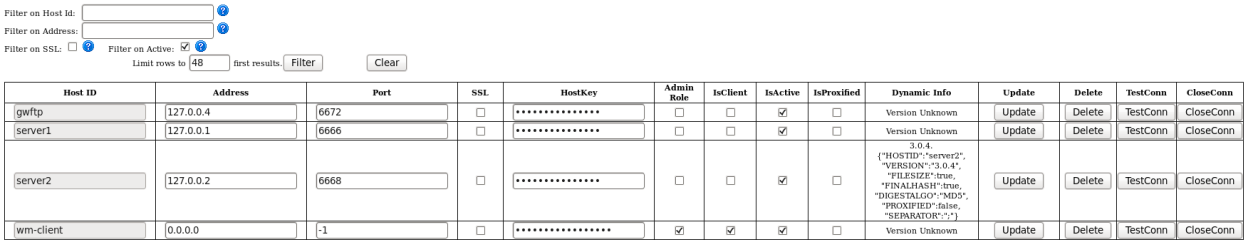
SpecialId	Rule	Filename	Info	TransferInfo	Step (LastStep)	Action	Status	Internal	Block	isMoved	Requester	Requested	Start	Stop	Cancel / Stop / Restart
-9223372036854775798	Rule Name:CONFIG_DIFF_SERVER2_2 MODETRANS: SENDMODE	/in/-9223372036854775798_3391523674041728069_server2-20161129180310.zip	noinfo	{"ORIGINALSIZE":1405}	POSTTASK (POSTTASK)	0	Current step in running **NotActive**	DONE : Transfer step ok	1	false	server1-ssl	server2-ssl	2016-11-29 18:03:23.62	2017-02-02 17:37:22.924	Cancel CancelClean Stop Restart
-9223372036854775801	Rule Name:CONFIG_DIFF_GWFTP_3 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/gwftp-20161129180310.zip	noinfo	{"ORIGINALSIZE":1542}	ERRORTASK (NOTASK)	0	Connection Impossible **NotActive**	INERROR : Connection Impossible	0	false	server2-ssl	gwftp-ssl	2016-11-29 18:03:21.76	2016-11-29 18:03:56.211	Cancel CancelClean Stop Restart
-9223372036854775800	Rule Name:CONFIG_DIFF_GWFTP_2 MODETRANS: SENDMODE	/in/-9223372036854775800_4831282629661183737_gwftp-20161129180310.zip	noinfo	{"ORIGINALSIZE":1542}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	1	false	server1-ssl	server2-ssl	2016-11-29 18:03:21.526	2016-11-29 18:03:21.816	Cancel CancelClean Stop Restart
-9223372036854775802	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/aaa	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	false	server2-ssl	gwftp-ssl	2016-02-19 15:07:46.394	2016-02-19 15:07:49.77	Cancel CancelClean Stop Restart
-9223372036854775802	Rule Name:toFTP MODETRANS: SENDMODE	/in/-9223372036854775802_5678557295443102166_aaa	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	false	server1	server2	2016-02-19 15:07:46.147	2016-02-19 15:07:46.445	Cancel CancelClean Stop Restart
-9223372036854775803	Rule Name:toFTP_2 MODETRANS: SENDMODE	/var/lib/waarp/server2/out/gwftp/test1.file	noinfo	{"ORIGINALSIZE":1048323}	ALLDONETASK (ALLDONETASK)	0	Operation completed **NotActive**	DONE : Operation completed	16	false	server2-ssl	gwftp-ssl	2016-02-19 15:07:12.986	2016-02-19 15:07:15.35	Cancel CancelClean Stop Restart

4.2.2 Moniteur

Accéder à l'interface des moniteurs



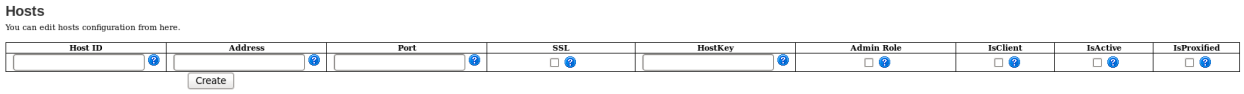
Lister les hôtes



Un filtrage est possible sur :

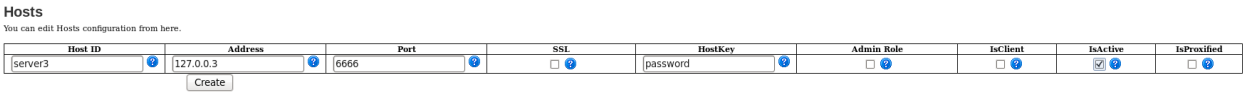
- l'hostid
- l'addresse IP
- l'utilisation de SSL
- l'état Actif du moniteur

Création d'un hôte

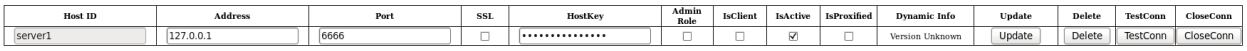


Les champs décrivant un hôtes sont les suivants :

- Hostname
- Address
- Port
- SSL
- HostKey
- Admin Role
- IsClient
- IsActive
- IsProxified

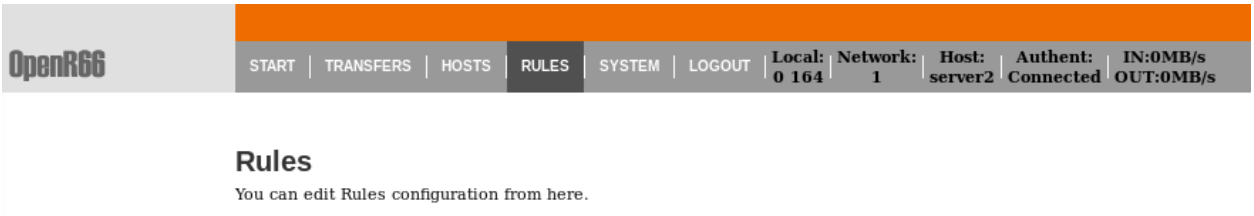


Mis à jour d'un hôte

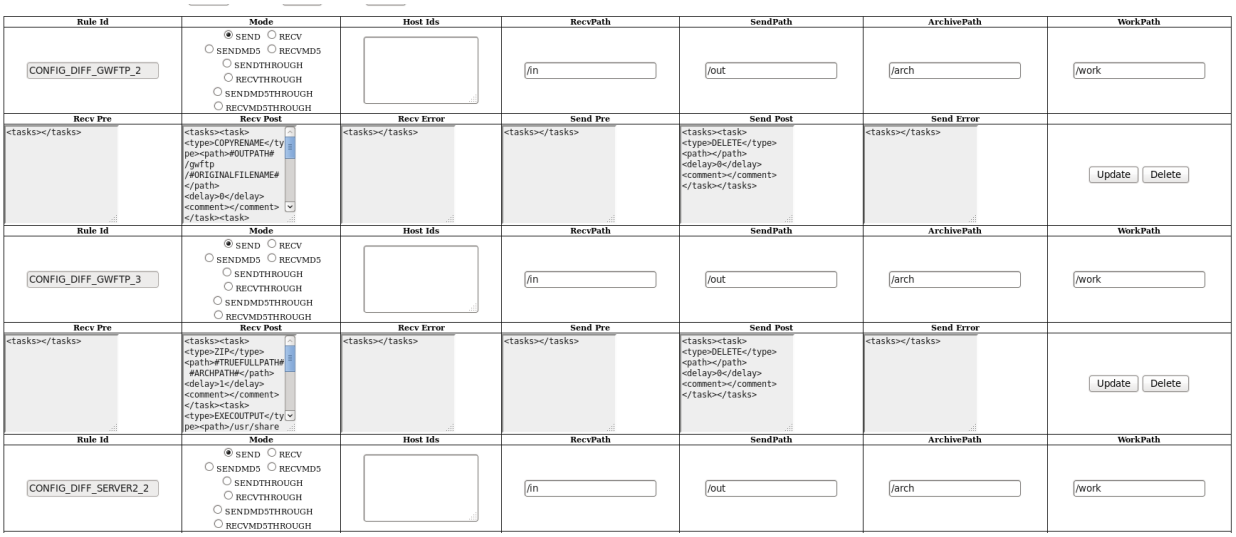


4.2.3 Règle

Accéder à l'interface des règles



Listing des règles

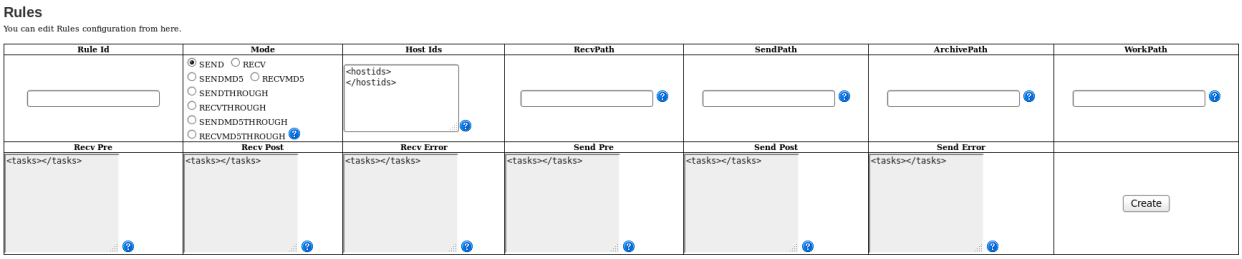


Un filtrage est possible sur :

- le nom de la règle
- le type de règle

Création de règle

Depuis l'interface d'adminsitration il est possible de créer de nouvelles règles de transferts



Une règle est composé de son ID, nom de la règle (celui ci doit être unique). D'un mode d'envoie

- SEND transfert d'envoie, le fichier va du client au serveur
- RECV transfert de reception, le fichier va du serveur au client

Ces 2 commandes peuvent etre completer par les :

- « » transfert classique
- MD5 transfert avec verification d'intégrite par packet
- THROUGH
- MD5THROUGH

Ex : SEND, RECVMD5, SENDTHROUGH

Hostids liste la liste des moniteurs authorisés à utiliser la règle.

RecvPath, SendPath, ArchPath, WorkPath sont respectivement les dossiers, de reception, d'envoie, d'archive et de travail de la règle.

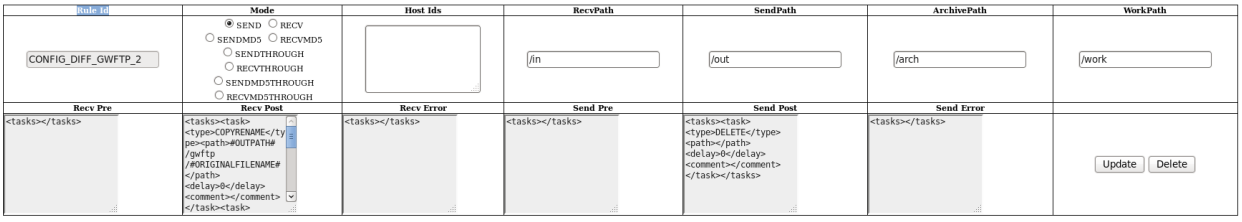
Les RecvPre, RecvPost et RecvError sont les tâches effectuées par le moniteur recepteur respectivement avant le transfert, apres le transfert et si le transfert tombre en erreur. Même chose pour les SendPre, SendPost, et SendError pour le moniteur emmeteur.

Les tâche éxécutées par les rêgles sont décrites comme suit :

- Type de tâche
- Path, utilisé pour spécifier les options de la tâche
- Delay, temps aloué avant échec
- When, condition d'execution

Pour plus de détails référez vous à la liste des tâches.

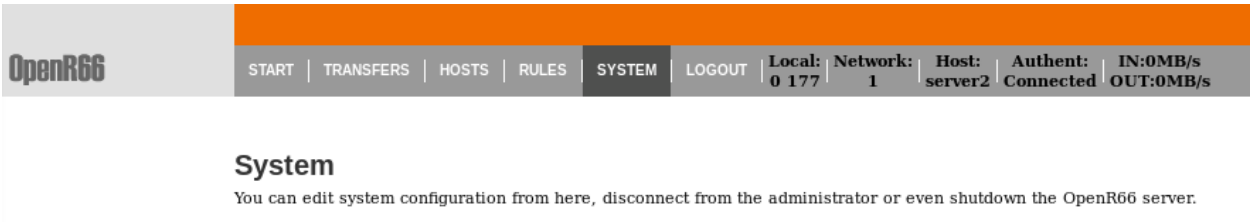
Mis à jour de règle



4.2.4 Type de tâche

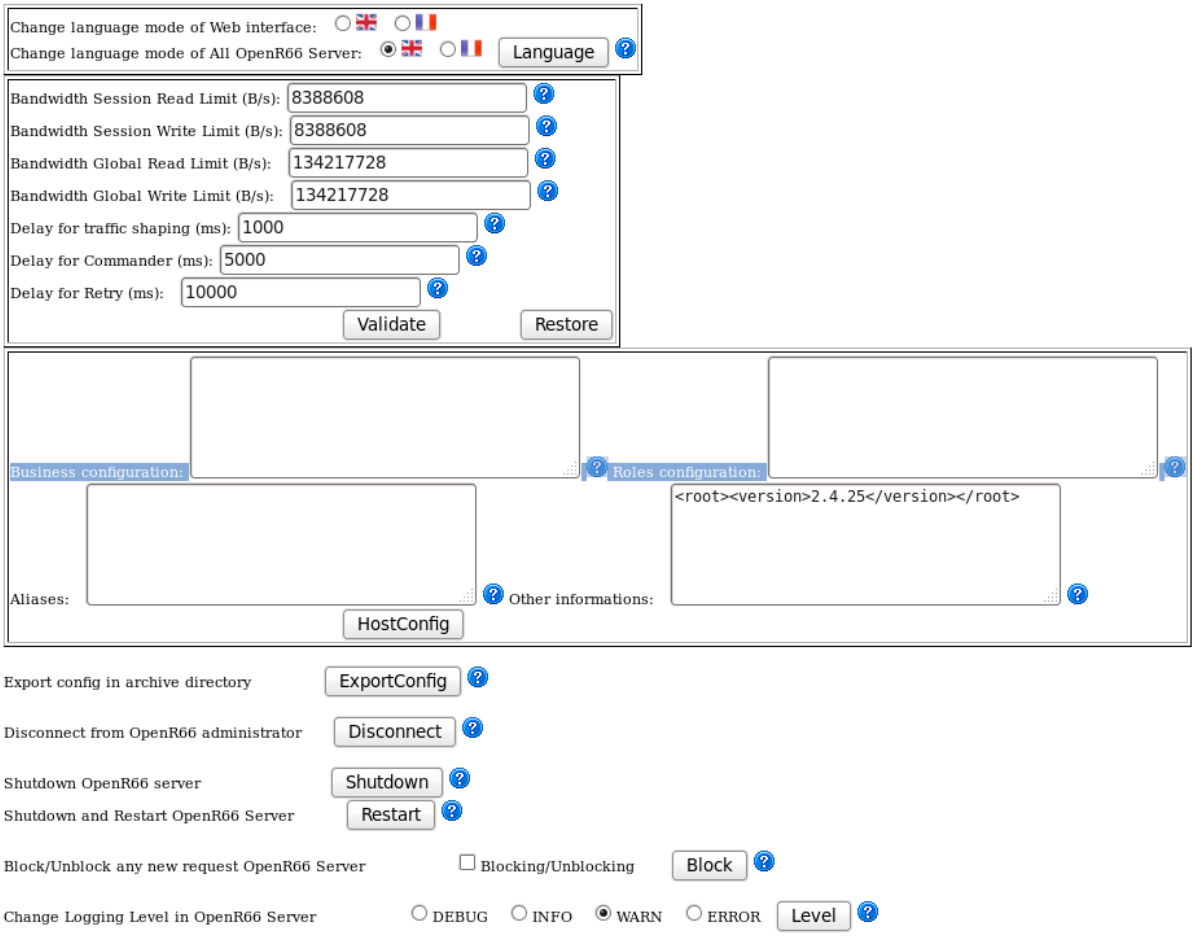
Nom	Path	Description
LOG		
MOVE	Dossier de destinaiton du fichier	Déplace le fichier
MOVERENAME	Dossier/nom de destination du fichier	Déplace et renomme le fichier
COPY	Dossier de destination de la copie	Copie le fichier
COPYRENAME	Dossier/nom de destination de la copie	Copie et renomme le fichier
EXEC	Accès au binaire à executer	Execute une programme tiers
EXECMOVE	Accès au binaire à executer	
EXECOUTPUT	Accès au binaire à executer	Execute un programme et log la sortie
EXECJAVA	Classe java à executer	Execute une classe Java
TRANSFER	Détails du nouveau transfert	Lance un nouveau transfert R66
VALIDFILEPATH		
DELETE		Supprime le fichier
LINKRENAME		
RESCHEDULE		Planifie une relance du transfert
TAR		Crée une archive TAR du fichier
ZIP		Crée une archive ZIP du fichier
TRANSCODE	Détails de la transcodification	Change le code page du fichier
SNMP		
FTP	Détails du transfert FTP	Envoie le fichier a un serveur FTP

4.2.5 Système



L'écran système vous permet de

— Changer la langue de l'interface d'administration
— Editer les limitations d'utilisation de la bande passante
— Editer les configurations annexes du moniteur (business, rôle, allias, . . .)



4.3 REST

4.3.1 Généralités

Authentification des requêtes

L'authentification des requêtes doit être activée dans la configuration du serveur Waarp R66 (valeur de la balise `restauthenticated` à `true`).

Si la requête doit être authentifiée, le nom de l'utilisateur doit être ajouté à la requête en définissant l'en-tête HTTP *X-Auth-User*.

Note : Le mot de passe n'est jamais transmis en clair dans la requête, mais sert à générer la *signature de la requête*.

Pour plus de sécurité, il vaut mieux activer la signature quand les requêtes sont authentifiées.

Signature des requêtes

La signature des requêtes REST permet de s'assurer que les requêtes n'ont pas été modifiées entre l'envoi et l'émission du message http (*man in the middle*) en transmettant une signature cryptographique de la requête.

La signature des requêtes doit être activée dans la configuration du serveur Waarp R66 (valeur de la balise `restsignature` à `true`).

Si la requête doit être signée, la signature doit être ajoutée à la requête en définissant l'en-tête HTTP *X-Auth-Key*. La valeur de cet en-tête est calculé selon l'algorithme décrit ci-dessous.

Calcul de la signature

La signature est calculée à partir de du chemin et de la chaîne de requête de l'URL requêtée selon l'algorithme suivant :

1. Tous les arguments de la chaîne de requête (*query string*), auxquels sont ajoutés les entêtes de `X-Auth-Timestamp` et `X-Auth-User` si ceux-ci sont utilisés, sont convertis en minuscules et triés dans l'ordre alphabétique des identifiants pour reconstruire une nouvelle chaine de requête.
2. Le paramètre `X-Auth-InternalKey` est ajouté à la chaîne de requête obtenue à l'étape precedente, avec pour valeur le mot de passe de l'utilisateur.
3. La chaîne de caractère utilisée pour le calcul de la signature est construite en concaténant le chemin de l'url et la chaîne de requête obtenue précédemment.
4. Enfin, la signature est calculée en appliquant la fonction HMAC-SHA256 à la chaîne de caractère obtenue à l'étape précédente avec la clef secrète du serveur (définie dans la balise `restsigkey` de sa configuration). C'est la représentation hexadécimale de la signature qui est conservée.

Exemple

On effectue une requête pour exporter l'historique des transferts terminés vers un fichier XML. La requête est envoyée à l'URL `http://127.0.0.1:8088/log` avec l'utilisateur `adminuser` (mot de passe : `adminpass`). La conficuration du serveur demande à ce que la requête soit horodatée.

La chaîne de requete utilisée pour le calcul de la signature sera donc `x-auth-user=adminuser&x-auth-timestamp=2017-04-12T23:52Z`.

Après ajout du mot de passe (étape 2), elle devient `x-auth-user=adminuser&x-auth-timestamp=2017-04-12T23:20:50.52Z&X-Auth-InternalKey=adminpass`.

La chaine de caractère complète utilisée pour le calcul de la signature est donc `/log?x-auth-user=adminuser&x-auth-timestamp=2017-04-12T23:20:50.52Z&X-Auth-InternalKey=adminpass`.

Enfin, c'est la représentation hexadécimale du hash HMAC-SHA256 appliqué à cette chaîne de caractère qui est utilisée comme signature.

Clef de signature

La clef utilisée pour la génération de la signature est stockée dans un fichier dont le chemin doit être renseigné dans la balise `restsigkey` du fichier de configuration du serveur.

Il s'agit d'une séquence aléatoire de 32 octets. Sous Linux, elle peut être générée avec la commande suivante :

```
# Pour afficher une clef
head -b 32 /dev/urandom

# Pour générer directement le fichier
head -b 32 /dev/urandom > path/to/restsigning.key
```

Horodatage des requêtes

Pour augmenter la sécurité de l'API, un serveur peut demander à ce que les requêtes soient horodatées.

L'horodatage des requêtes doit être activé dans la configuration du serveur Waarp R66 (valeur de la balise `resttimelimit` supérieur à `0`).

La valeur de la balise `resttimelimit` représente la durée de validité de la requête : si l'écart entre l'heure du serveur Waarp R66 au moment de la recéption de la requête et la date contenue dans la requête est supérieur à `resttimelimit`, la requête est refusée.

Si la requête doit être horodatée, le date et l'heure doivent être ajoutées à la requête en définissant l'en-tête HTTP *X-Timestamp*. La date doit être au format **RFC 3339** (exemple : `2018-03-22T16:00:05.352Z`).

4.3.2 Gestion de l'historique

Export et purge de l'historique

`GET /log`

Ce point d'entrée permet d'exporter l'historique de transfert du serveur Waarp R66 associé au client, et le cas échéant de purger l'historique.

Les fichiers XML produit sont déposés dans le dossier arch définitions dans la configuration du serveur (le chemin complet est fourni dans la réponse de la requête).

Paramètres de la requête

Le corps de la réponse doit être un objet JSON valide. Celui-ci continet plusieurs groupes de paramètres.

Les paramètres `@class` et `requestUserPacket` sont obligatoires et leur valeur est fixe (voir ci-dessous).

Le second groupe de paramètres permet de filtrer l'historique exporté par date, statut, identifiant, règle de transfert et/ou partenaire.

Enfin, deux paramètres permettent d'effectuer des opérations de maintenance conjointement à l'export : `purge`, qui supprime de la base de données l'historique exporté, et `clean`, qui corrige le statut de transferts erronés quand celui-ci est erroné.

Request JSON Object

- **`@class`** (*`string`*) – Le type de requête. Doit être `org.waarp.openr66.protocol.localhandler.packet.json.LogJsonPacket`
- **`comment`** (*`string`*) – Un commentaire optionnel
- **`requestUserPacket`** (*`int`*) – Le type de requête. Doit être `16`
- **`statuspending`** (*`bool`*) – Exporte les transferts en attente (défaut :`false`)
- **`statustransfer`** (*`bool`*) – Exporte les transferts en cours(défaut : `false`)
- **`statusdone`** (*`bool`*) – Exporte les transferts terminés (défaut : `false`)
- **`statuserror`** (*`bool`*) – Exporte les transferts en erreur (défaut : `false`)
- **`rule`** (*`string`*) – Limite l'export à une règle spécifique
- **`string request`** (*`int`*) – Corrige le statut des transferts terminés erronés
- **`start`** (*`int`*) – Exporte seulement l'historique postérieur à cette date. La date doit être fournie sous la forme d'un timestamp Unix *en millisecondes*.
- **`stop`** (*`int`*) – Exporte seulement l'historique postérieur à cette date. La date doit être fournie sous la forme d'un timestamp Unix *en millisecondes*.
- **`startid`** (*`int`*) – Valeur minimale d'identifiants de transfert à exporter
- **`stopid`** (*`int`*) – Valeur maximale d'identifiants de transfert à exporter
- **`purge`** (*`bool`*) – Si `true`, l'historique exporté est également purgé de la base de données (defaut : `false`)
- **`clean`** (*`bool`*) – Corrige le statut des transferts terminés erronés (defaut : `false`)

Détails de la réponse

La réponse contient le statut de la requête, ainsi que de nombreuses données récapitulant la requête.

Les éléments les plus significatifs de la réponse sont les suivants :

Response JSON Object

- **answer.results.0.filename** (*string*) – Le chemin complet du fichier contenant les données exportées
- **answer.results.0.exported** (*int*) – Le nombre de transferts exportés
- **answer.results.0.purged** (*int*) – Le nombre de transferts purgés
- **message** (*string*) – Statut de la requête comme texte (OK signifie un succès)
- **code** (*int*) – Statut de la requête comme code réponse HTTP
- **details** (*string*) – En cas d'erreur (code retour différent de 200, un message expliquant la cause de l'erreeur

Codes retours

Les requêtes vers ce point d'entrée peuvent avoir les code retour HTTP suivants. En cas d'erreur, les détails peuvent se retrouver dans le champ `details` de la réponse.

Status Codes

- 200 OK – Succès
- 400 Bad Request – Une erreur est présente dans le corps de la requête
- 401 Unauthorized – Authentification invalide pour l'utilisateur
- 405 Method Not Allowed – Le point d'entrée est désactivé pour ce serveur

Exemple de requête

```
GET /log HTTP/1.1
Content-Type: application/json

{
   "@class": "org.waarp.openr66.protocol.localhandler.packet.json.LogJsonPacket",
   "requestUserPacket": 16,
   "purge": false,
   "stop": 1399760601400
}
```

Exemple de réponse (succès)

```
HTTP/1.1 200 OK
content-type: application/json

{
   "X-method":"GET",
   "path":"/log",
   "base":"log",
   "uri":{},
   "answer":{
      "@model":"Log",
      "results":[{
         "@class":"org.waarp.openr66.protocol.localhandler.packet.json.
↪LogResponseJsonPacket",
         "comment":null,
         "requestUserPacket":16,
         "purge":false,
         "clean":false,
         "statuspending":false,
         "statustransfer":false,
         "statusdone":false,
         "statuserror":false,
```

(suite sur la page suivante)

(suite de la page précédente)


```
            "rule":null,
            "request":null,
            "start":null,
            "stop":1399760601400,
            "startid":null,
            "stopid":null,
            "command":16,
            "filename":"[...]/data/server1/arch/server1_1521715697441_runners.xml",
            "exported":0,
            "purged":0
        }]
    },
    "command":"GetLog",
    "message":"OK",
    "code":200
}
```


Exemple de réponse (erreur)


```
HTTP/1.1 400 Bad Request
content-type: application/json

{
    "code": 400,
    "detail": "com.fasterxml.jackson.databind.JsonMappingException:
        Unexpected token (END_OBJECT), expected FIELD_NAME: missing
        property '@class' that is to contain type id  (for class
        org.waarp.openr66.protocol.localhandler.packet.json.JsonPacket)\n
        at [Source: {\"class\":
        \"org.waarp.openr66.protocol.localhandler.packet.json.LogJsonPacket\",
        \"requestUserPacket\": 16, \"purge\": true, \"stop\":
        1399760601400}; line: 1, column: 141]",
    "message": "Bad Request"
}
```


4.3.3 /transfers

GET /transfers

Retourne les transferts monitorés par le moniteur

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbTaskRunner]

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple


```
[{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
}]
```

Paramètres

Paramètre	Type	Desctiption
LIMIT	integer	Maximum number of transfers returned
ORDERBYID	bool	Is the answer sorted by ID
STARTID	integer	Lower transfer ID returned
STOPID	integer	Higher transfer ID returned
IDRULE	string	Name of the rule used
PARTNER	string	Hostname of transfer partner
PENDING	bool	Return « Pending » transfer
INTRANSFER	bool	Return « Running » transfer
INERROR	bool	Return « Failed » transfer
DONE	bool	Return « Completed » transfer
ALLSTATUS	bool	Return all transfer
STARTTRANS	ISO 8061 or ms	Return transfer after this time
STOPSTRANS	ISO 8061 or ms	Return transfer before this time
OWNERREQ	string	Owner of the request

GET /transfers/ :id

Retourne le transfert spécifié

Réponse

Modèle

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple


```
{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
}
```

Paramètres

Paramètre	Type	Desctiption
SPECIALID	integer	Special Id as LONG in URI as transfers/id
REQUESTER	varchar	Partner as requester
REQUESTED	varchar	Partner as requested
OWNERREQ	varchar	Owner of this request

POST /transfers

Ajoute un nouveau transfer au moniteur

Réponse

Modèle

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple


```
{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
}
```

Paramètres

Paramètre	Type	Desctiption
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	varchar	
RETRIEVEMODE	bit	
FILENAME	varchar	
ISMOVED	bit	
IDRULE	varchar	
BLOCKSZ	integer	
ORIGINALNAME	varchar	
FILEINFO	longvarchar	
TRANSFERINFO	longvarchar	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	varchar	
UPDATEDINFO	integer	
OWNERREQ	varchar	
REQUESTER	varchar	
REQUESTED	varchar	
SPECIALID	bigint	

PUT /transfers/ :id

Modifie le transfert spécifié

Réponse

Modèle

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple

```
{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
}
```

Paramètres

Paramètre	Type	Desctiption
SPECIALID	bigint	
REQUESTER	varchar	
REQUESTED	varchar	
OWNERREQ	varchar	
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	varchar	
RETRIEVEMODE	bit	
FILENAME	varchar	
ISMOVED	bit	
BLOCKSZ	integer	
ORIGINALNAME	varchar	
FILEINFO	longvarchar	
TRANSFERINFO	longvarchar	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	varchar	
UPDATEDINFO	integer	

DELETE /transfers/ :id

Supprime le transfert spécifié

Réponse

Modèle

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple

```
{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
}
```

Paramètres

Paramètre	Type	Desctiption
SPECIALID	integer	Special Id as LONG in URI as transfers/id
REQUESTER	varchar	Partner as requester
REQUESTED	varchar	Partner as requested
OWNERREQ	varchar	Owner of this request

4.3.4 /hosts

GET /hosts

Retourne tous les hôtes connus du moniteur

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbHostAuth]

DbHostAuth

Champs	Type	Description
ADDRESS	string	
PORT	integer	
ISSSL	bit	
HOSTKEY	[binary]	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
[{
  "ADDRESS" : "127.0.0.1",
  "PORT" : "6666",
  "ISSSL" : 0,
  "HOSTKEY" : "VARBINARY",
  "ADMINROLE" : false,
  "ISCLIENT" : false,
  "ISACTIVE" : true,
  "ISPROXIFIED" : false,
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "server1"
},]
```

Paramètres

Paramètre	Type	Desctiption
HOSTID	string	Host name
ADDRESS	string	Address of this partner
ISSSL	bool	Is Ssl entry
ISACTIVE	bool	Is Active entry

GET /hosts/ :id

Retourne l'hôte spécifié

Réponse

Modèle

DbHostAuth

Champs	Type	Description
ADDRESS	string	
PORT	integer	
ISSSL	bit	
HOSTKEY	[binary]	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "ADDRESS" : "127.0.0.1",
  "PORT" : "6666",
  "ISSSL" : 0,
  "HOSTKEY" : "VARBINARY",
  "ADMINROLE" : false,
  "ISCLIENT" : false,
  "ISACTIVE" : true,
  "ISPROXIFIED" : false,
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "server1"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	string	HostId in URI as hosts/id

POST /hosts

Crée un nouvel hôte

Réponse

Modèle

DbHostAuth

Champs	Type	Description
ADDRESS	string	
PORT	integer	
ISSSL	bit	
HOSTKEY	[binary]	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "ADDRESS" : "127.0.0.1",
  "PORT" : "6666",
  "ISSSL" : 0,
  "HOSTKEY" : "VARBINARY",
  "ADMINROLE" : false,
  "ISCLIENT" : false,
  "ISACTIVE" : true,
  "ISPROXIFIED" : false,
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "server1"
}
```

Paramètres

Paramètre	Type	Desctiption
ADDRESS	varchar	
PORT	integer	
ISSSL	bit	
HOSTKEY	varbinary	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	varchar	

PUT /hosts/ :id

Modifie l'hôte spécifié

Réponse

Modèle

DbHostAuth

Champs	Type	Description
ADDRESS	string	
PORT	integer	
ISSSL	bit	
HOSTKEY	[binary]	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "ADDRESS" : "127.0.0.1",
  "PORT" : "6666",
  "ISSSL" : 0,
  "HOSTKEY" : "VARBINARY",
  "ADMINROLE" : false,
  "ISCLIENT" : false,
  "ISACTIVE" : true,
  "ISPROXIFIED" : false,
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "server1"
}
```

Paramètres

Paramètre	Type	Desctiption
HOSTID	varchar	HostId in URI as hosts/id
ADDRESS	varchar	
PORT	integer	
ISSSL	bit	
HOSTKEY	varbinary	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	

DELETE /hosts/ :id

Supprime l'hôte spécifié

Réponse

Modèle

DbHostAuth

Champs	Type	Description
ADDRESS	string	
PORT	integer	
ISSSL	bit	
HOSTKEY	[binary]	
ADMINROLE	bit	
ISCLIENT	bit	
ISACTIVE	bit	
ISPROXIFIED	bit	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
{
  "ADDRESS" : "127.0.0.1",
  "PORT" : "6666",
  "ISSSL" : 0,
  "HOSTKEY" : "VARBINARY",
  "ADMINROLE" : false,
  "ISCLIENT" : false,
  "ISACTIVE" : true,
  "ISPROXIFIED" : false,
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "server1"
}
```


Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hosts/id

4.3.5 /hostconfigs

GET /hostconfigs

Retourne toutes les configurations des hôotes

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbHostConfiguration]

DbHostConfiguration

Champs	Type	Description
BUSINESS	string	
ROLES	string	
ALIASES	string	
OTHERS	string	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
[{
  "BUSINESS" : "LONGVARCHAR",
  "ROLES" : "LONGVARCHAR",
  "ALIASES" : "LONGVARCHAR",
  "OTHERS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
},]
```


Paramètres

Paramètre	Type	Desctiption
BUSINESS	longvarchar	
ROLES	longvarchar	
ALIASES	longvarchar	
OTHERS	longvarchar	
HOSTID	varchar	

GET /hostconfigs/ :id

Réponse

Modèle

DbHostConfiguration

Champs	Type	Description
BUSINESS	string	
ROLES	string	
ALIASES	string	
OTHERS	string	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
{
  "BUSINESS" : "LONGVARCHAR",
  "ROLES" : "LONGVARCHAR",
  "ALIASES" : "LONGVARCHAR",
  "OTHERS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id

PUT /hostconfigs/ :id

Réponse

Modèle

DbHostConfiguration

Champs	Type	Description
BUSINESS	string	
ROLES	string	
ALIASES	string	
OTHERS	string	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "BUSINESS" : "LONGVARCHAR",
  "ROLES" : "LONGVARCHAR",
  "ALIASES" : "LONGVARCHAR",
  "OTHERS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id
BUSINESS	longvarchar	
ROLES	longvarchar	
ALIASES	longvarchar	
OTHERS	longvarchar	
UPDATEDINFO	integer	

POST /hostconfigs

Réponse

Modèle

DbHostConfiguration

Champs	Type	Description
BUSINESS	string	
ROLES	string	
ALIASES	string	
OTHERS	string	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "BUSINESS" : "LONGVARCHAR",
  "ROLES" : "LONGVARCHAR",
  "ALIASES" : "LONGVARCHAR",
  "OTHERS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
BUSINESS	longvarchar	
ROLES	longvarchar	
ALIASES	longvarchar	
OTHERS	longvarchar	
UPDATEDINFO	integer	
HOSTID	varchar	

DELETE /hostconfigs/ :id

Réponse

Modèle

DbHostConfiguration

Champs	Type	Description
BUSINESS	string	
ROLES	string	
ALIASES	string	
OTHERS	string	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "BUSINESS" : "LONGVARCHAR",
  "ROLES" : "LONGVARCHAR",
  "ALIASES" : "LONGVARCHAR",
  "OTHERS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id

4.3.6 /configurations

GET /configurations

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DBConfiguration]

DbConfiguration

Champs	Type	Description
READGLOBALLIMIT	integer	
WRITEGLOBALLIMIT	integer	
READSESSIONLIMIT	integer	
WRITESESSIONLIMIT	integer	
DELAYLIMIT	integer	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
[{
  "READGLOBALLIMIT" : "BIGINT",
   "WRITEGLOBALLIMIT" : "BIGINT",
   "READSESSIONLIMIT" : "BIGINT",
   "WRITESESSIONLIMIT" : "BIGINT",
   "DELAYLIMIT" : "BIGINT",
   "UPDATEDINFO" : "INTEGER",
   "HOSTID" : "VARCHAR"
},]
```


Paramètres

Paramètre	Type	Desctiption
HOSTID	string	host name
BAND-WIDTH	inte-ger	<0 for no filter, =0 for no bandwidth, >0 for a limit greater than value integer

GET /configurations/ :id

Réponse

Modèle

DbConfiguration

Champs	Type	Description
READGLOBALLIMIT	integer	
WRITEGLOBALLIMIT	integer	
READSESSIONLIMIT	integer	
WRITESESSIONLIMIT	integer	
DELAYLIMIT	integer	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
{
  "READGLOBALLIMIT" : "BIGINT",
   "WRITEGLOBALLIMIT" : "BIGINT",
   "READSESSIONLIMIT" : "BIGINT",
   "WRITESESSIONLIMIT" : "BIGINT",
   "DELAYLIMIT" : "BIGINT",
   "UPDATEDINFO" : "INTEGER",
```


(suite sur la page suivante)

(suite de la page précédente)

```
    "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id

POST /configurations

Réponse

Modèle

DbConfiguration

Champs	Type	Description
READGLOBALLIMIT	integer	
WRITEGLOBALLIMIT	integer	
READSESSIONLIMIT	integer	
WRITESESSIONLIMIT	integer	
DELAYLIMIT	integer	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "READGLOBALLIMIT" : "BIGINT",
  "WRITEGLOBALLIMIT" : "BIGINT",
  "READSESSIONLIMIT" : "BIGINT",
  "WRITESESSIONLIMIT" : "BIGINT",
  "DELAYLIMIT" : "BIGINT",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
READGLOBALLIMIT	bigint	
WRITEGLOBALLIMIT	bigint	
READSESSIONLIMIT	bigint	
WRITESESSIONLIMIT	bigint	
DELAYLIMIT	bigint	
updatedinfo	integer	
HOSTID	varchar	

PUT /configurations/ :id

Réponse

Modèle

DbConfiguration

Champs	Type	Description
READGLOBALLIMIT	integer	
WRITEGLOBALLIMIT	integer	
READSESSIONLIMIT	integer	
WRITESESSIONLIMIT	integer	
DELAYLIMIT	integer	
UPDATEDINFO	integer	
HOSTID	string	

Exemple


```
{
  "READGLOBALLIMIT" : "BIGINT",
  "WRITEGLOBALLIMIT" : "BIGINT",
  "READSESSIONLIMIT" : "BIGINT",
  "WRITESESSIONLIMIT" : "BIGINT",
  "DELAYLIMIT" : "BIGINT",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id
READGLOBALLIMIT	bigint	
WRITEGLOBALLIMIT	bigint	
READSESSIONLIMIT	bigint	
WRITESESSIONLIMIT	bigint	
DELAYLIMIT	bigint	
updatedinfo	integer	

DELETE /configurations/ :id

Réponse

Modèle

DbConfiguration

Champs	Type	Description
READGLOBALLIMIT	integer	
WRITEGLOBALLIMIT	integer	
READSESSIONLIMIT	integer	
WRITESESSIONLIMIT	integer	
DELAYLIMIT	integer	
UPDATEDINFO	integer	
HOSTID	string	

Exemple

```
{
  "READGLOBALLIMIT" : "BIGINT",
  "WRITEGLOBALLIMIT" : "BIGINT",
  "READSESSIONLIMIT" : "BIGINT",
  "WRITESESSIONLIMIT" : "BIGINT",
  "DELAYLIMIT" : "BIGINT",
  "UPDATEDINFO" : "INTEGER",
  "HOSTID" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Desctiption
:id	varchar	HostId in URI as hostconfigs/id

4.3.7 /rules

GET /rules

Réponse

Modèle

Champs	Type

DbRule

Champs	Type	Description
HOSTIDS	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	
IDRULE	string	

Exemple

```
[{
  "HOSTIDS" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "RECVPATH" : "VARCHAR",
  "SENDPATH" : "VARCHAR",
  "ARCHIVEPATH" : "VARCHAR",
  "WORKPATH" : "VARCHAR",
  "RPRETASKS" : "LONGVARCHAR",
  "RPOSTTASKS" : "LONGVARCHAR",
  "RERRORTASKS" : "LONGVARCHAR",
  "SPRETASKS" : "LONGVARCHAR",
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "SPOSTTASKS" : "LONGVARCHAR",
  "SERRORTASKS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "IDRULE" : "VARCHAR"
},]
```

Paramètres

Paramètre	Type	Description
« IDRULE »		rule name
« MODETRANS »		MODETRANS value

GET /rules/ :id

Réponse

Modèle

DbRule

Champs	Type	Description
HOSTIDS	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	
IDRULE	string	

Exemple

```
{
  "HOSTIDS" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "RECVPATH" : "VARCHAR",
  "SENDPATH" : "VARCHAR",
  "ARCHIVEPATH" : "VARCHAR",
  "WORKPATH" : "VARCHAR",
  "RPRETASKS" : "LONGVARCHAR",
```

(suite sur la page suivante)

(suite de la page précédente)

```
  "RPOSTTASKS" : "LONGVARCHAR",
  "RERRORTASKS" : "LONGVARCHAR",
  "SPRETASKS" : "LONGVARCHAR",
  "SPOSTTASKS" : "LONGVARCHAR",
  "SERRORTASKS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "IDRULE" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Description
IDRULE	string	RuleId in URI as rules/id

POST /rules

Réponse

Modèle

DbRule

Champs	Type	Description
HOSTIDS	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	
IDRULE	string	

Exemple


```
{
  "HOSTIDS" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "RECVPATH" : "VARCHAR",
  "SENDPATH" : "VARCHAR",
  "ARCHIVEPATH" : "VARCHAR",
  "WORKPATH" : "VARCHAR",
  "RPRETASKS" : "LONGVARCHAR",
  "RPOSTTASKS" : "LONGVARCHAR",
  "RERRORTASKS" : "LONGVARCHAR",
  "SPRETASKS" : "LONGVARCHAR",
  "SPOSTTASKS" : "LONGVARCHAR",
  "SERRORTASKS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "IDRULE" : "VARCHAR"
}
```


Paramètres

Paramètre	Type	Description
HOSTIDS »	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	

PUT /rules/ :id

Réponse

Modèle

DbRule

Champs	Type	Description
HOSTIDS	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	
IDRULE	string	

Exemple

```
{
  "HOSTIDS" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "RECVPATH" : "VARCHAR",
  "SENDPATH" : "VARCHAR",
  "ARCHIVEPATH" : "VARCHAR",
  "WORKPATH" : "VARCHAR",
  "RPRETASKS" : "LONGVARCHAR",
  "RPOSTTASKS" : "LONGVARCHAR",
  "RERRORTASKS" : "LONGVARCHAR",
  "SPRETASKS" : "LONGVARCHAR",
  "SPOSTTASKS" : "LONGVARCHAR",
  "SERRORTASKS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "IDRULE" : "VARCHAR"
}
```

Paramètres

Paramètre	Type	Description
IDRULE	string	RuleId in URI as rules/id
HOSTIDS »	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	

DELETE /rules/ :id

Réponse

Modèle

DbRule

Champs	Type	Description
HOSTIDS	string	
MODETRANS	integer	
RECVPATH	string	
SENDPATH	string	
ARCHIVEPATH	string	
WORKPATH	string	
RPRETASKS	string	
RPOSTTASKS	string	
RERRORTASKS	string	
SPRETASKS	string	
SPOSTTASKS	string	
SERRORTASKS	string	
UPDATEDINFO	integer	
IDRULE	string	

Exemple


```
{
  "HOSTIDS" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "RECVPATH" : "VARCHAR",
  "SENDPATH" : "VARCHAR",
  "ARCHIVEPATH" : "VARCHAR",
  "WORKPATH" : "VARCHAR",
  "RPRETASKS" : "LONGVARCHAR",
  "RPOSTTASKS" : "LONGVARCHAR",
  "RERRORTASKS" : "LONGVARCHAR",
  "SPRETASKS" : "LONGVARCHAR",
  "SPOSTTASKS" : "LONGVARCHAR",
  "SERRORTASKS" : "LONGVARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "IDRULE" : "VARCHAR"
}
```


Paramètres

Paramètre	Type	Description
IDRULE	string	RuleId in URI as rules/id

4.3.8 /info

GET /info

Réponse

Modèle

Champs	Type
answer	Array[path]

Exemple


```
[
  "PathA",
  "PathB"
]
```

Paramètre

Paramètre	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.InformationJsonPacket
comment	string	Information request (GET)
requestUserPacket	integer	18
id	integer	0
request	integer	0
rulename	string	The rule name associated with the remote repository
filename	string	The filename to look for if any
idRequest	boolean	false
to	boolean	false

GET /info

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbTaskRunner]

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	integer	

Exemple


```
[{
  "GLOBALSTEP" : "INTEGER",
  "GLOBALLASTSTEP" : "INTEGER",
  "STEP" : "INTEGER",
  "RANK" : "INTEGER",
  "STEPSTATUS" : "VARCHAR",
  "RETRIEVEMODE" : "BIT",
  "FILENAME" : "VARCHAR",
  "ISMOVED" : "BIT",
  "IDRULE" : "VARCHAR",
  "BLOCKSZ" : "INTEGER",
  "ORIGINALNAME" : "VARCHAR",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : "INTEGER",
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
},]
```


Paramètre

Paramètre	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.InformationJsonPacket
comment	string	Information on Transfer request (GET)
requestUserPacket	integer	18
id	integer	0
request	integer	0
rulename	string	remoteHost
filename	string	null
idRequest	boolean	true
to	boolean	false

4.3.9 /bandwidth

GET /bandwidth

Réponse

Modèle

Champs	Type
answer	Array[DbBandwith]

DbBandwith

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.BandwidthJsonPacket
comment	string	Bandwidth getter (GET)
requestUserPacket	integer	19
setter	boolean	false
writeglobal	integer	-10
readglobal	integer	-10
writesession	integer	-10
readsession	integer	-10

Exemple

```
[{
  "@class": "org.waarp.openr66.protocol.localhandler.packet.json.
↪BandwidthJsonPacket",
  "comment": "Bandwidth getter (GET)",
  "requestUserPacket": 19,
  "setter": false,
  "writeglobal": -10,
  "readglobal": -10,
  "writesession": -10,
  "readsession": -10
}]
```

Paramètres

Paramètre	Type	Description
@class	string	
comment	string	
requestUserPacket	string	
setter	boolean	
writeglobal	integer	
readglobal	integer	
writesession	integer	
readsession	integer	

PUT /bandwidth

Réponse

Modèle

Champs	Type
answer	Array[DbBandwith]

DbBandwith

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.BandwidthJsonPacket
comment	string	Bandwidth setter (SET)
requestUserPacket	integer	19
setter	boolean	false
writeglobal	integer	-10
readglobal	integer	-10
writesession	integer	-10
readsession	integer	-10

Exemple

```
[{
  "@class" : "org.waarp.openr66.protocol.localhandler.packet.json.
↪BandwidthJsonPacket",
  "comment" : "Bandwidth setter (SET)",
  "requestUserPacket" : 19,
  "setter" : true,
  "writeglobal" : -10,
  "readglobal" : -10,
  "writesession" : -10,
  "readsession" : -10
},]
```

Paramètres

Paramètre	Type	Description
@class	string	
comment	string	
requestUserPacket	string	
setter	boolean	
writeglobal	integer	
readglobal	integer	
writesession	integer	
readsession	integer	

4.3.10 /control

GET /control

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbTaskRunner]

DbTaskRunner

Champs	Type	Description
GLOBALSTEP	integer	
GLOBALLASTSTEP	integer	
STEP	integer	
RANK	integer	
STEPSTATUS	string	
RETRIEVEMODE	bit	
FILENAME	string	
ISMOVED	bit	
IDRULE	string	
BLOCKSZ	integer	
ORIGINALNAME	string	
FILEINFO	string	
TRANSFERINFO	string	
MODETRANS	integer	
STARTTRANS	timestamp	
STOPTRANS	timestamp	
INFOSTATUS	string	
UPDATEDINFO	integer	
OWNERREQ	string	
REQUESTER	string	
REQUESTED	string	
SPECIALID	int	

Exemple


```
[{
  "GLOBALSTEP" : 1,
  "GLOBALLASTSTEP" :2,
  "STEP" : 1,
  "RANK" : 1,
  "STEPSTATUS" : "PENDING",
  "RETRIEVEMODE" : 1,
  "FILENAME" : "test-file.txt",
  "ISMOVED" : 0,
  "IDRULE" : "2a1",
  "BLOCKSZ" : 32,
  "ORIGINALNAME" : "test-file.txt",
  "FILEINFO" : "LONGVARCHAR",
  "TRANSFERINFO" : "LONGVARCHAR",
  "MODETRANS" : 1,
  "STARTTRANS" : "TIMESTAMP",
  "STOPTRANS" : "TIMESTAMP",
  "INFOSTATUS" : "VARCHAR",
  "UPDATEDINFO" : "INTEGER",
  "OWNERREQ" : "VARCHAR",
  "REQUESTER" : "VARCHAR",
  "REQUESTED" : "VARCHAR",
  "SPECIALID" : "BIGINT"
},]
```


Paramètres

Paramètre	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.InformationJsonPacket
comment	string	Information on Transfer request (GET)
requestUserPacket	integer	18
id	integer	0
request	integer	0
rulename	string	remoteHost
filename	string	null
idRequest	boolean	true
to	boolean	false

PUT /control

Réponse

Modèle

Champs	Type
DbConfiguration	Array[DbTaskRunner]

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.RestartTransferJsonPacket
comment	string	Restart Transfer request (PUT)
requestUserPacket	integer	4
requester	string	Requester host
requested	string	Requested host
specialid	integer	-9223372036854775808
restarttime	integer	1399760601381

Exemple

```
[{
  "@class" : "org.waarp.openr66.protocol.localhandler.packet.json.
↪RestartTransferJsonPacket"
  "comment" : "Restart Transfer request (PUT)"
  "requestUserPacket" : 4
  "requester" : "Requester host"
  "requested" : "Requested host"
  "specialid" : -9223372036854775808
  "restarttime" : 1399760601381
},]
```

Paramètres

Paramètre	Type	Description
@class	string	Class to execute ({Restart,Cancel}TransferJsonPacket)
comment	string	Restart Transfer request (PUT)
requestUserPacket	integer	4
requester	string	Requester host
requested	string	Requested host
specialid	integer	-9223372036854775808
restarttime	timestamp	1399760601381

POST /control

Réponse

Modèle

Champs	Type

DbTaskRunner

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.TransferRequestJsonPacket
comment	string	Transfer Request (POST)
requestUserPacket	integer	7
rulename	string	Rulename
mode	integer	0
filename	string	Filename
requested	string	Requested host
blocksize	integer	0
rank	integer	0
specialId	integer	0
validate	integer	0
originalSize	integer	0
fileInformation	string	File information
separator	char	{
start	integer	1399760601381
delay	integer	0
toValidate	boolean	true
additionalDelay	boolean	false

Exemple

```
[{
  "@class" : "org.waarp.openr66.protocol.localhandler.packet.json.
↪TransferRequestJsonPacket",
  "comment" : "Transfer Request (POST)",
  "requestUserPacket" : 7,
  "rulename" : "Rulename",
  "mode" : 0,
  "filename" : "Filename",
  "requested" : "Requested host",
  "blocksize" : 0,
  "rank" : 0,
  "specialId" : 0,
  "validate" : 0,
```

(suite sur la page suivante)

(suite de la page précédente)


```
  "originalSize" : 0,
  "fileInformation" : "File information",
  "separator" : "{",
  "start" : 1399760601381,
  "delay" : 0,
  "toValidate" : true,
  "additionalDelay" : false
},]
```


Paramètres

Paramètre	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.TransferRequestJsonPacket
comment	string	Transfer Request (POST)
requestUserPacket	integer	7
rulename	string	Rulename
mode	integer	0
filename	string	Filename
requested	string	Requested host
blocksize	integer	0
rank	integer	0
specialId	integer	0
validate	integer	0
originalSize	integer	0
fileInformation	string	File information
separator	char	{
start	timestamp	1399760601381
delay	integer	0
toValidate	boolean	true
additionalDelay	boolean	false

4.3.11 /server

GET /server

Réponse

Modèle

Champs	Type
answer	Array[]

Champs	Type	Description
HostID	string	hosta
Date	timestamp	2014-05-13T11 :02 :52.185+02 :00
LastRun	timestamp	1970-01-01T01 :00 :00.000+01 :00
FromDate	timestamp	1970-01-01T01 :00 :00.000+01 :00
SecondsRunning	integer	0
NetworkConnections	integer	0
NbThreads	integer	0
InBandwidth	integer	0
OutBandwidth	integer	0
OVERALL	OVERALL	
STEPS	STEPS	
RUNNINGSTEPS	RUNNINGSTEPS	
ERRORTYPES	ERRORTYPES	

OVERALL

Champs	Type	Description
AllTransfer	integer	0
Unknown	integer	0
NotUpdated	integer	0
Interrupted	integer	0
ToSubmit	integer	0
Error	integer	0
Running	integer	0
Done	integer	0
InRunning	integer	0
OutRunning	integer	0
LastInRunning	timestamp	2014-05-13T11 :02 :43.732+02 :00
LastOutRunning	timestamp	2014-05-13T11 :02 :43.732+02 :00
InAll	integer	0
OutAll	integer	0
InError	integer	0
OutError	integer	0

STEPS

Champs	Type	Description
Notask	integer	0
Pretask	integer	0
Transfer	integer	0
Posttask	integer	0
AllDone	integer	0
Error	integer	0

RUNNINGSTEPS

Champs	Type	Description
AllRunning	integer	0
Running	integer	0
InitOk	integer	0
PreProcessingOk	integer	0
TransferOk	integer	0
PostProcessingOk	integer	0
CompleteOk	integer	0

ERRORTYPES

Champs	Type	Description
ConnectionImpossible	interger	0
ServerOverloaded	interger	0
BadAuthent	interger	0
ExternalOp	interger	0
TransferError	interger	0
MD5Error	interger	0
Disconnection	interger	0
FinalOp	interger	0
Unimplemented	interger	0
Internal	interger	0
Warning	interger	0
QueryAlreadyFinished	interger	0
QueryStillRunning	interger	0
KnownHost	interger	0
RemotelyUnknown	interger	0
CommandNotFound	interger	0
PassThroughMode	interger	0
RemoteShutdown	interger	0
Shutdown	interger	0
RemoteError	interger	0
Stopped	interger	0
Canceled	interger	0
FileNotFound	interger	0
Unknown	interger	0

Exemple


```
[{
  "HostID" : "hosta",
  "Date" : "2014-05-13T11:02:52.185+02:00",
  "LastRun" : "1970-01-01T01:00:00.000+01:00",
  "FromDate" : "1970-01-01T01:00:00.000+01:00",
  "SecondsRunning" : 0,
  "NetworkConnections" : 0,
  "NbThreads" : 0,
  "InBandwidth" : 0,
  "OutBandwidth" : 0,
  "OVERALL" : {
```


(suite sur la page suivante)

(suite de la page précédente)


```
    "AllTransfer" : 0,
    "Unknown" : 0,
    "NotUpdated" : 0,
    "Interrupted" : 0,
    "ToSubmit" : 0,
    "Error" : 0,
    "Running" : 0,
    "Done" : 0,
    "InRunning" : 0,
    "OutRunning" : 0,
    "LastInRunning" : "2014-05-13T11:02:43.732+02:00",
    "LastOutRunning" : "2014-05-13T11:02:43.732+02:00",
    "InAll" : 0,
    "OutAll" : 0,
    "InError" : 0,
    "OutError" : 0
  },
  "STEPS" : {
    "Notask" : 0,
    "Pretask" : 0,
    "Transfer" : 0,
    "Posttask" : 0,
    "AllDone" : 0,
    "Error" : 0
  },
  "RUNNINGSTEPS" : {
    "AllRunning" : 0,
    "Running" : 0,
    "InitOk" : 0,
    "PreProcessingOk" : 0,
    "TransferOk" : 0,
    "PostProcessingOk" : 0,
    "CompleteOk" : 0
  },
  "ERRORTYPES" : {
    "ConnectionImpossible" : 0,
    "ServerOverloaded" : 0,
    "BadAuthent" : 0,
    "ExternalOp" : 0,
    "TransferError" : 0,
    "MD5Error" : 0,
    "Disconnection" : 0,
    "FinalOp" : 0,
    "Unimplemented" : 0,
    "Internal" : 0,
    "Warning" : 0,
    "QueryAlreadyFinished" : 0,
    "QueryStillRunning" : 0,
    "KnownHost" : 0,
    "RemotelyUnknown" : 0,
    "CommandNotFound" : 0,
    "PassThroughMode" : 0,
```


(suite sur la page suivante)

(suite de la page précédente)


```
    "RemoteShutdown" : 0,
    "Shutdown" : 0,
    "RemoteError" : 0,
    "Stopped" : 0,
    "Canceled" : 0,
    "FileNotFound" : 0,
    "Unknown" : 0
  }
},]
```


PUT /server

Réponse

Modèle

Champs	Type
answer	Array[]

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.ShutdownOrBlockJsonPacket
comment	string	Shutdown Or Block request (PUT)
requestUserPacket	integer	0
key	[byte]	S2V5
shutdownOrBlock	boolean	false
restartOrBlock	boolean	false

Exemple


```
[{
  "@class" : "org.waarp.openr66.protocol.localhandler.packet.json.
↪ShutdownOrBlockJsonPacket",
  "comment" : "Shutdown Or Block request (PUT)",
  "requestUserPacket" : 0,
  "key" : "S2V5",
  "shutdownOrBlock" : false,
  "restartOrBlock" : false
},]
```

Paramètres

Paramètre	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.ShutdownOrBlockJsonPacket
comment	string	Shutdown Or Block request (PUT)
requestUserPacket	integer	0
key	[byte]	S2V5
shutdownOrBlock	boolean	false
restartOrBlock	boolean	false

4.3.12 /business

GET /business

Réponse

Modèle

Champs	Type
answer	Array[]

Champs	Type	Description
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.BusinessRequestJsonPacket
comment	string	Business execution request (GET)
requestUserPacket	integer	22
className	string	Class name to execute
arguments	string	Arguments of the execution
extraArguments	string	Extra arguments
delay	integer	0
toApplied	boolean	false
validated	boolean	false

Exemple

```
[{
  "@class" : "org.waarp.openr66.protocol.localhandler.packet.json.
↪BusinessRequestJsonPacket"
  "comment" : "Business execution request (GET)",
  "requestUserPacket" : 22,
  "className" : "Class name to execute",
  "arguments" : "Arguments of the execution",
  "extraArguments" : "Extra arguments",
  "delay" : 0,
  "toApplied" : false,
  "validated" : false
},]
```

Paramètres

Paramètre	Type	Desctiption
@class	string	org.waarp.openr66.protocol.localhandler.packet.json.BusinessRequestJsonPacket
comment	string	Business execution request (GET)
requestUserPacket	integer	22
className	string	Class name to execute
arguments	string	Arguments of the execution
extraArguments	string	Extra arguments
delay	integer	0
toApplied	boolean	false
validated	boolean	false

4.4 REST v2

ATTENTION : L'API REST v2 est actuellement en beta et est donc susceptible de contenir des bugs.

Cette documentation ayant pour but d'être courte et concise, en conséquence elle peut parfois manquer de précision. En cas de besoin, une documentation un peu plus exhaustive peut être consultée ici.

Une specification détaillée écrite en RAML peut également être téléchargée si nécessaire en utilisant `ce lien` ou au format OpenAPI 3.0 `ce lien`.

4.4.1 Authentification des requêtes

Lorsque l'authentification des requêtes est activée dans la configuration REST du serveur, celle-ci peut se faire de 2 manières différentes.

Authentification Basique

Une requête peut-être authentifiée avec l'authentification basique. Il s'agit de l'authentification HTTP basique standard telle qu'elle est définie dans la RFC 2617.

Cette méthode d'authentification a l'avantage d'être simple a utilisée, mais elle n'est pas sûre sans utilisation de SSL car les identifiants utilisateur seront envoyés en clair sur le réseau. Il n'est donc recommandé d'utiliser cette méthode uniquement en combinaison avec HTTPS.

Cette authentification se fait avec les entêtes suivants :

- **Authorization** - Cet entête doit contenir la chaîne de caractères "Basic", afin d'annoncer l'utilisation du mode d'authentification basique, suivi du nom d'utilisateur et de son mot de passe. Le nom d'utilisateur et le mot de passe doivent être séparés par le caractère deux-points, et le tout doit être encodé en Base64.
  - *exemple* : Pour l'utilisateur "toto" avec le mot de passe "totomdp" cela donne
    -> `Basic toto:totomdp` avant encodage
    -> `Basic dG90bzp0b3RvbWRw` après encodage

Authentification HMAC

Une requête REST peut également être authentifiée en utilisant un hash des identifiants utilisateur.

Cette méthode à l'avantage d'être sûre, même sans utilisation de SSL, mais elle nécessite que le client ait connaissance au préalable de la clé de signature REST du serveur. Cette clé est la clé renseignée dans la configuration REST du serveur.

Cette authentification se fait avec les entêtes suivants :

- **X-Auth-User** - Cet entête contient le nom de l'utilisateur s'authentifiant.
- **X-Auth-Timestamp** - Cet entête contient la date (en format ISO 8601) d'émission de la requête. Cette date est utilisée pour déterminer si la requête a expirée ou non. La durée de validité d'une requête est fixée dans la configuration REST du serveur.
- **Authorization** - Cet entête contient le hash des identifiants utilisateur. Ce hash est obtenu avec l'algorithme de hash *HMAC SHA256* en utilisant la clé de signature REST du serveur. La chaîne de caractères originale est la concaténation des éléments suivants :
  - la date de la requête (i.e. l'entête *X-Auth-Timestamp*)
  - le nom d'utilisateur (i.e. l'entête *X-Auth-User*)
  - le mot de passe de l'utilisateur

  Le hash doit être préfixé de la chaîne "HMAC" pour spécifier l'utilisation du mode d'authentication HMAC.
  - *exemple* : Pour l'utilisateur "toto" avec le mot de passe "totomdp" à la date "1970-01-01T01 :00 :00+00 :00" cela donne
    -> `HMAC 1970-01-01T01:00:00+00:00tototototomdp` avant hachage
    -> `HMAC e4219167eb4cf1f8590d684713218c4ad011d475d8f3b2d37fb15ce3da675021` après hachage

4.4.2 Signature des requêtes

Pour s'assurer que le contenu d'une requête n'est pas modifié lors de son acheminement, les requêtes peuvent être signées avec un hash de leur contenu. La signature de requête nécessite que l'authentification des requêtes soit également activée sur le serveur.

La signature des requêtes se fait avec les entêtes suivants :

— **X-Auth-Signature** - Cet entête contient un hash du contenu de la requête. Ce hash est obtenu avec l'algorithme de hash *HMAC SHA256* en utilisant la clé de signature REST du serveur. La chaîne de caractères originale est la concaténation des éléments suivants :
  — les identifiants de l'utilisateur (i.e. l'entête *Authorization*)
  — le corps de la requête
  — l'URI de la requête
  — la méthode HTTP de la requête

  Cela permet de s'assurer qu'aucun de ces éléments n'a été altéré durant sa transmission.
  — *exemple* : Pour l'utilisateur "toto" avec le mot de passe "totomdp" envoyant une requête "PUT" sur l'URI "/v2/hosts/serveur1" avec le contenu

```
{ "port": 8080 }
```

  cela donne
  -> `Basic toto:totomdp{ "port": 8080 }/v2/hosts/serveur1PUT` avant encodage et hachage
  -> `Basic dG90bzp0b3RvbWRw{ "port": 8080 }/v2/hosts/serveur1PUT` après encodage
  -> `4dd8bfd9c1c537dbe67ce1572ceac0c18fbce70b2d08100ebd1fe773c32573dd` après hachage

4.4.3 Gestion des transferts

Lister les transferts

Nouveau dans la version 3.4.0 : Ajout du paramètre `followId`

GET /v2/transfers

Renvoie une liste des transferts monitorés par le serveur respectant les filtres donnés en paramètres de requête, ou simplement le décompte si countOrder=true est passé en paramètre.

Query Parameters

— **limit** (*integer*) – Nombre maximal de transferts pouvant être inclus dans la réponse.
— **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse.
— **order** (*string*) – Ordre dans lequel les réponses seront triées. Valeurs possibles : *ascId*, *descId*, *ascFile*, *descFile*, *ascStart*, *descStart*, *ascStop*, *descStop*
— **ruleName** (*string*) – Filter les transferts par règle de transfert.
— **partner** (*string*) – Filtrer les transferts par partenaire.
— **status** (*string*) – Filtrer les transferts par statut. Valeurs possibles : *TOSUBMIT*, *NOTUPDATED*, *RUNNING*, *INTERRUPTED*, *DONE*, *INERROR*, *UNKNOWN*
— **filename** (*string*) – Filtrer les transferts par fichier.
— **startTrans** (*string*) – Filtrer les transferts commençant après cette date (format ISO-8601).
— **stopTrans** (*string*) – Filtrer les transferts commençant avant cette date (format ISO-8601).
— **followId** (*string*) – Identifiant FollowId à rechercher dans les transferts.
— **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Example request :

```
GET /v2/transfers HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.
Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "totalResults": 1,
    "results": [
        {
            "id": 1,
            "globalStep": "NOTASK",
            "globalLastStep": "NOTASK",
            "step": 1,
            "rank": 1,
            "status": "TOSUBMIT",
            "stepStatus": "string",
            "originalFilename": "string",
            "filename": "string",
            "ruleName": "string",
            "blockSize": 1,
            "fileInfo": "string",
            "transferInfo": "string",
            "start": "2021-08-18T12:47:27.252876",
            "stop": "2021-08-18T12:47:27.252876",
            "requester": "string",
            "requested": "string"
        }
    ]
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
```

(suite sur la page suivante)

(suite de la page précédente)

```
}
```

Ajouter un transfert

POST /v2/transfers

Initialise un nouveau transfert sur le serveur avec les attributs définis dans le corps de la requête. Les informations sur le transfert créé sont renvoyées dans la réponse ainsi que l'URI pour le consuter.

Example request :

```
POST /v2/transfers HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "ruleName": "default",
    "filename": "example_file.txt",
    "requested": "server2",
    "blockSize": 65536,
    "fileInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00"
}
```

Status Codes

— 201 Created – Le nouveau transfert a été créé avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "id": 1,
    "globalStep": "NOTASK",
    "globalLastStep": "NOTASK",
    "step": 0,
    "rank": 0,
    "status": "TOSUBMIT",
    "stepStatus": "",
    "originalFilename": "out/file_example.txt",
    "filename": "in/file_example.txt",
    "ruleName": "default",
    "blockSize": 65536,
    "fileInfo": "{\"ORIGINALSIZE\":1024}",
    "transferInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00",
    "stop": "1970-01-01T02:00:00+00:00",
    "requester": "server1",
    "requested": "server2"
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

Response Headers

— **transferURI** – L'URI d'accès pour consulter le transfert.

Consulter un transfert

GET /v2/transfers/{id}

Si le transfert demandé existe, renvoie toutes ses informations.

Parameters

— **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Example request :

```
GET /v2/transfers/{id} HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "id": 1,
    "globalStep": "NOTASK",
    "globalLastStep": "NOTASK",
    "step": 0,
    "rank": 0,
    "status": "TOSUBMIT",
    "stepStatus": "",
    "originalFilename": "out/file_example.txt",
```

(suite sur la page suivante)

(suite de la page précédente)

```
    "filename": "in/file_example.txt",
    "ruleName": "default",
    "blockSize": 65536,
    "fileInfo": "{\"ORIGINALSIZE\":1024}",
    "transferInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00",
    "stop": "1970-01-01T02:00:00+00:00",
    "requester": "server1",
    "requested": "server2"
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Arrêter un transfert

PUT /v2/transfers/{id}/stop

S'il existe, arrête le transfert demandé.

Parameters

— **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

— 202 Accepted – Le transfert a été arrêté avec succès.

Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "id": 1,
    "globalStep": "NOTASK",
    "globalLastStep": "NOTASK",
    "step": 0,
    "rank": 0,
    "status": "TOSUBMIT",
    "stepStatus": "",
    "originalFilename": "out/file_example.txt",
    "filename": "in/file_example.txt",
    "ruleName": "default",
    "blockSize": 65536,
    "fileInfo": "{\"ORIGINALSIZE\":1024}",
    "transferInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00",
    "stop": "1970-01-01T02:00:00+00:00",
    "requester": "server1",
    "requested": "server2"
}
```

Redémarrer un transfert

PUT /v2/transfers/{id}/restart

S'il existe, redémarre le transfert demandé.

Parameters

— **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

— 202 Accepted – Le transfert a été redémarré avec succès.

Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "id": 1,
    "globalStep": "NOTASK",
    "globalLastStep": "NOTASK",
    "step": 0,
    "rank": 0,
    "status": "TOSUBMIT",
    "stepStatus": "",
    "originalFilename": "out/file_example.txt",
    "filename": "in/file_example.txt",
    "ruleName": "default",
    "blockSize": 65536,
    "fileInfo": "{\"ORIGINALSIZE\":1024}",
    "transferInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00",
    "stop": "1970-01-01T02:00:00+00:00",
    "requester": "server1",
    "requested": "server2"
}
```

Annuler un transfert

PUT /v2/transfers/{id}/cancel

If it exists, cancels the requested transfer.

Parameters

— **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

— 202 Accepted – Le transfert a été annulé avec succès.

Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json
```

(suite sur la page suivante)

(suite de la page précédente)


```
{
    "id": 1,
    "globalStep": "NOTASK",
    "globalLastStep": "NOTASK",
    "step": 0,
    "rank": 0,
    "status": "TOSUBMIT",
    "stepStatus": "",
    "originalFilename": "out/file_example.txt",
    "filename": "in/file_example.txt",
    "ruleName": "default",
    "blockSize": 65536,
    "fileInfo": "{\"ORIGINALSIZE\":1024}",
    "transferInfo": "This is comment example.",
    "start": "1970-01-01T01:00:00+00:00",
    "stop": "1970-01-01T02:00:00+00:00",
    "requester": "server1",
    "requested": "server2"
}
```


4.4.4 Gestion des hôtes

Lister les hôtes

GET /v2/hosts

Renvoie l'ensemble des hôtes conformes aux filtres passés en paramètres de requête, ou simplement le décompte si countOrder=true est passé en paramètre.. Les hôtes peuvent être filtrés par addresse, sur leur utilisation de SSL, ou sur le fait qu'ils soient actifs ou non. Les paramètres *offset* et *limit* permettent de fixer un numéro de départ et un nombre maximal d'hôtes désirés, afin de ne renvoyer qu'un sous-ensemble de la liste demandée. Le paramètre *order* spécifie dans quel ordre les hôtes doivent être ordonées.

Query Parameters

- **limit** (*integer*) – Fixe le nombre maximal de transferts pouvant être inclus dans la réponse. Utile lorsque le nombre d'entrées sélectionnées par les filtres est trop important pour tenir dans un seule message.
- **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse. Généralement utilisé en combinaison avec le paramètre *limit* pour restreindre l'ensemble des hôtes à renvoyer.
- **order** (*string*) – Spécifie l'attribut à utiliser pour trier les éléments de la réponse, ainsi que le sens du tri. Ce paramètre devrait donc être le nom d'un attribut d'hôte préfixé par « asc » ou « desc », correspondant respectivement à l'ordre croissant et décroissant.
- **address** (*string*) – Si ce paramètre est défini, la réponse ne contiendra que les hôtes ayant cette adresse.
- **isSSL** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que les hôtes autorisant les transferts SSL. Et si le paramètre est faux, que ceux ne les autorisant pas. Si le paramètre n'est pas défini, aucun filtrage ne sera fait.
- **isActive** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que les hôtes actifs. Et si le paramètre est faux, que ceux inactifs. Si le paramètre n'est pas défini, aucun filtrage ne sera fait.
- **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Example request :

```
GET /v2/hosts HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.
Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "totalResults": 1,
    "results": [
        {
            "name": "string",
            "address": "string",
            "port": 1,
            "password": "string",
            "isSSL": true,
            "isAdmin": true,
            "isClient": true,
            "isActive": true,
            "isProxy": true
        }
    ]
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

— 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

Ajouter un hôte

`POST /v2/hosts`

Ajoute un nouvel hôte à la base de données avec les attribut de l'objet passé dans le corps de la requête. La nouvelle entrée créée est renvoyée dans la réponse, ainsi que l'URI pour la consulter.

Example request :

```
POST /v2/hosts HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "name": "server1",
    "address": "127.0.0.1",
    "port": 8080,
    "password": "password",
    "isSSL": false,
    "isAdmin": false,
    "isClient": false,
    "isActive": true,
    "isProxy": false
}
```

Status Codes

— 201 Created – Le nouvel hôte a été ajouté avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "name": "server1",
    "address": "127.0.0.1",
    "port": 8080,
    "password": "password",
    "isSSL": false,
    "isAdmin": false,
    "isClient": false,
    "isActive": true,
    "isProxy": false
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

— 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

Response Headers

— **hostURI** – L'URI d'accès aux informations du nouvel hôte.

Consulter un hôte

GET /v2/hosts/{id}

S'il existe, renvoie les informations de l'hôte demandé sous forme d'un objet JSON.

Example request :

```
GET /v2/hosts/{id} HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – L'hôte demandé a été trouvé.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "name": "server1",
    "address": "127.0.0.1",
    "port": 8080,
    "password": "password",
    "isSSL": false,
    "isAdmin": false,
    "isClient": false,
    "isActive": true,
    "isProxy": false
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Modifier un hôte

PUT /v2/hosts/{id}

S'il existe, remplace les informations de l'hôte demandé par celles données dans la requête. Les nouvelles informations seront renvoyées dans la réponse, ainsi que leur URI d'accès.

Example request :

```
PUT /v2/hosts/{id} HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "name": "server1",
    "address": "127.0.0.1",
    "port": 8080,
    "password": "password",
    "isSSL": false,
    "isAdmin": false,
    "isClient": false,
    "isActive": true,
    "isProxy": false
}
```

Status Codes

— 202 Accepted – L'hôte a été mis à jour avec succès.

Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "name": "server1",
    "address": "127.0.0.1",
    "port": 8080,
    "password": "password",
    "isSSL": false,
    "isAdmin": false,
    "isClient": false,
    "isActive": true,
    "isProxy": false
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.
— 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

Response Headers

— **host-uri** – Le chemin d'accès de l'hôte modifié. Si le nom d'hôte n'a pas été changé, cet URI sera identique à celui ayant été utilisé pour la requête.

Supprimer un hôte

DELETE /v2/hosts/{id}

S'il existe, supprime l'hôte de la base de données.

Parameters

— **id** (*string*) – Le nom unique de l'hôte désiré.

Status Codes

— 204 No Content – L'hôte a été supprimé avec succès.
— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

4.4.5 Gestion des règles

Lister les règles

GET /v2/rules

Renvoie l'ensemble des règles de transfert conformes aux filtres passés en paramètres de requête, ou simplement le décompte si countOrder=true est passé en paramètre. Les règles peuvent être filtrées par mode de transmission. Les paramètres *offset* et *limit* permettent de fixer un numéro de départ et un nombre maximal de règles désirées, afin de ne renvoyer qu'un sous-ensemble de la liste demandée. Le paramètre *order* spécifie dans quel ordre les règles doivent être ordonées.

Query Parameters

— **limit** (*integer*) – Fixe le nombre maximal de règles pouvant être inclues dans la réponse. Utile lorsque le nombre d'entrées sélectionnées par les filtres est trop important pour tenir dans un seule message.
— **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse. Généralement utilisé en combinaison avec le paramètre *limit* pour restreindre l'ensemble des hôtes à renvoyer. **Attention** : Si le *offset* est plus grand que le nombre total de réponses, aucune entrée ne sera renvoyée.

— **order** (*string*) – Spécifie l'attribut à utiliser pour trier les éléments de la réponse, ainsi que le sens du tri. Ce paramètre devrait donc être le nom d'un attribut d'hôte préfixé par « asc » ou « desc », correspondant respectivement à l'ordre croissant et décroissant.
— **modeTrans** (*string*) – Si ce paramètre est défini, la réponse ne contiendra que les règles utilisant ce mode de transfert.
— **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Example request :

```
GET /v2/rules HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "totalResults": 1,
    "results": [
        {
            "name": "string",
            "hostIds": [
                "string"
            ],
            "modeTrans": "send",
            "recvPath": "string",
            "sendPath": "string",
            "archivePath": "string",
            "workPath": "string",
            "rPreTasks": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
            ],
            "rPostTask": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
            ],
            "rErrorTasks": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
```

(suite sur la page suivante)

(suite de la page précédente)


```
            ],
            "sPreTasks": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
            ],
            "sPostTasks": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
            ],
            "sErrorTasks": [
                {
                    "type": "LOG",
                    "arguments": "string",
                    "delay": 1
                }
            ]
        }
    ]
}
```


— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :


```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

Ajouter une règle

POST /v2/rules

Creates a new transfer rule with the ObjectRule passed in the request body. The created entry is returned in the response body, along with its URI.

Example request :

```
POST /v2/rules HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "name": "example",
    "hostIds": [
        "server1",
        "server2"
    ],
    "modeTrans": "send",
    "recvPath": "in/",
    "sendPath": "out/",
    "archivePath": "arch/",
    "workPath": "work/",
    "rPreTasks": [
        {
            "type": "ZIP",
            "arguments": "file.txt work/",
            "delay": 1
        }
    ],
    "rPostTask": [
        {
            "type": "MOVERENAME",
            "arguments": "arch/file.bak"
        }
    ],
    "rErrorTasks": [
        {
            "type": "RESCHEDULE",
            "arguments": "-delay 3600000 -case ConnectionImpossible,
↪ServerOverloaded,Shutdown"
        }
    ],
    "sPreTasks": [],
    "sPostTasks": [
        {
            "type": "ZIP",
            "arguments": "file.zip work/",
            "delay": 0
        }
    ],
    "sErrorTasks": [
        {
            "type": "DELETE"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        }
    ]
}
```

Status Codes

— 201 Created – La nouvelle règle de transfert a été ajoutée avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "name": "example",
    "hostIds": [
        "server1",
        "server2"
    ],
    "modeTrans": "send",
    "recvPath": "in/",
    "sendPath": "out/",
    "archivePath": "arch/",
    "workPath": "work/",
    "rPreTasks": [
        {
            "type": "ZIP",
            "arguments": "file.txt work/",
            "delay": 1
        }
    ],
    "rPostTask": [
        {
            "type": "MOVERENAME",
            "arguments": "arch/file.bak"
        }
    ],
    "rErrorTasks": [
        {
            "type": "RESCHEDULE",
            "arguments": "-delay 3600000 -case ConnectionImpossible,
↪ServerOverloaded,Shutdown"
        }
    ],
    "sPreTasks": [],
    "sPostTasks": [
        {
            "type": "ZIP",
            "arguments": "file.zip work/",
            "delay": 0
        }
    ],
    "sErrorTasks": [
        {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "type": "DELETE"
        }
    ]
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

Response Headers
— **ruleURI** – L'URI d'accès à la nouvelle règle créée.

Consulter une règle

GET /v2/rules/{id}
Si elle existe, renvoie les informations de la règle demandée sous forme d'un objet JSON.
Example request :

```
GET /v2/rules/{id} HTTP/1.1
Host: example.com
```

Status Codes
— 200 OK – La règle remandée a été trouvée.
Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "name": "example",
    "hostIds": [
        "server1",
```

(suite sur la page suivante)

(suite de la page précédente)

```
        "server2"
    ],
    "modeTrans": "send",
    "recvPath": "in/",
    "sendPath": "out/",
    "archivePath": "arch/",
    "workPath": "work/",
    "rPreTasks": [
        {
            "type": "ZIP",
            "arguments": "file.txt work/",
            "delay": 1
        }
    ],
    "rPostTask": [
        {
            "type": "MOVERENAME",
            "arguments": "arch/file.bak"
        }
    ],
    "rErrorTasks": [
        {
            "type": "RESCHEDULE",
            "arguments": "-delay 3600000 -case ConnectionImpossible,
↪ServerOverloaded,Shutdown"
        }
    ],
    "sPreTasks": [],
    "sPostTasks": [
        {
            "type": "ZIP",
            "arguments": "file.zip work/",
            "delay": 0
        }
    ],
    "sErrorTasks": [
        {
            "type": "DELETE"
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Modifier une règle

PUT /v2/rules/{id}

Si elle existe, remplace les informations de la règle demandée par celles données dans la requête. Les nouvelles informations seront renvoyées dans la réponse, ainsi que leur URI d'accès.

Example request :

```
PUT /v2/rules/{id} HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "name": "example",
    "hostIds": [
        "server1",
        "server2"
    ],
    "modeTrans": "send",
    "recvPath": "in/",
    "sendPath": "out/",
    "archivePath": "arch/",
    "workPath": "work/",
    "rPreTasks": [
        {
            "type": "ZIP",
            "arguments": "file.txt work/",
            "delay": 1
        }
    ],
    "rPostTask": [
        {
            "type": "MOVERENAME",
            "arguments": "arch/file.bak"
        }
    ],
    "rErrorTasks": [
        {
            "type": "RESCHEDULE",
            "arguments": "-delay 3600000 -case ConnectionImpossible,
↪ServerOverloaded,Shutdown"
        }
    ],
    "sPreTasks": [],
    "sPostTasks": [
        {
            "type": "ZIP",
            "arguments": "file.zip work/",
            "delay": 0
        }
    ],
    "sErrorTasks": [
        {
            "type": "DELETE"
```

(suite sur la page suivante)

(suite de la page précédente)

```
        }
    ]
}
```

Status Codes

— 201 Created – La règle de transfert a été mise à jour avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "name": "example",
    "hostIds": [
        "server1",
        "server2"
    ],
    "modeTrans": "send",
    "recvPath": "in/",
    "sendPath": "out/",
    "archivePath": "arch/",
    "workPath": "work/",
    "rPreTasks": [
        {
            "type": "ZIP",
            "arguments": "file.txt work/",
            "delay": 1
        }
    ],
    "rPostTask": [
        {
            "type": "MOVERENAME",
            "arguments": "arch/file.bak"
        }
    ],
    "rErrorTasks": [
        {
            "type": "RESCHEDULE",
            "arguments": "-delay 3600000 -case ConnectionImpossible,
↪ServerOverloaded,Shutdown"
        }
    ],
    "sPreTasks": [],
    "sPostTasks": [
        {
            "type": "ZIP",
            "arguments": "file.zip work/",
            "delay": 0
        }
    ],
    "sErrorTasks": [
        {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "type": "DELETE"
        }
    ]
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Response Headers

— **ruleURI** – Le chemin d'accès de la règle modifiée. Si le nom de la règle n'a pas été changé, cet URI sera identique à celui ayant été utilisé pour la requête.

Supprimer une règle

DELETE /v2/rules/{id}

Si elle existe, supprime la règle de transfert de la base de données.

Parameters

— **id** (*string*) – L'identifiant unique de la règle souhaitée.

Status Codes

— 204 No Content – La règle a été supprimée avec succès.

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

4.4.6 Gestion de la configuration

Initialiser la configuration

POST /v2/hostconfig

Initialise la configuration du serveur dans la base de données avec les attribut passés dans la requête.

Example request :

```
POST /v2/hostconfig HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
            "hostName": "server1",
            "roleList": [
                "TRANSFER",
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```

Status Codes

— 201 Created – La configuration du serveur a été ajoutée avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
            "hostName": "server1",
```

(suite sur la page suivante)

(suite de la page précédente)


```
            "roleList": [
                "TRANSFER",
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```


— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :


```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```


— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Consulter la configuration

GET /v2/hostconfig

Renvoie la configuration du serveur sous forme d'un objet JSON.

Example request :

```
GET /v2/hostconfig HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
            "hostName": "server1",
            "roleList": [
                "TRANSFER",
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Modifier la configuration

PUT /v2/hostconfig

Si le serveur a une configuration en base de données, remplace cette configuration avec les nouveaux attributs passé dans la requête.

Example request :

```
PUT /v2/hostconfig HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
            "hostName": "server1",
            "roleList": [
                "TRANSFER",
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```

Status Codes

— 202 Accepted – La configuration a été mise à jour avec succès.

Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
            "hostName": "server1",
            "roleList": [
                "TRANSFER",
```

(suite sur la page suivante)

(suite de la page précédente)

```
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Supprimer la configuration

DELETE /v2/hostconfig

Si le serveur a une configuration en base de données, supprime cette configuration.

Status Codes

— 204 No Content – La configuration a été supprimée avec succès.
— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

4.4.7 Gestion de la bande passante

Modifier les limites

POST /v2/limits

Initialise les limites de bande passante du serveur avec les attributs passés dans la requête.

Example request :

```
POST /v2/limits HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "upGlobalLimit": 1000000000,
    "downGlobalLimit": 50000000,
    "upSessionLimit": 1000000,
    "downSessionLimit": 500000,
    "delayLimit": 1000
}
```

Status Codes

— 201 Created – Les limites ont été initialisées avec succès.

Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "upGlobalLimit": 1000000000,
    "downGlobalLimit": 50000000,
    "upSessionLimit": 1000000,
    "downSessionLimit": 500000,
    "delayLimit": 1000
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.

Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "errorCode": 4
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Consulter les limites

GET /v2/limits

Renvoie les limites de bande passante sous forme d'un objet JSON.

Example request :

```
GET /v2/limits HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "upGlobalLimit": 1000000000,
    "downGlobalLimit": 50000000,
    "upSessionLimit": 1000000,
    "downSessionLimit": 500000,
    "delayLimit": 1000
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Modifier les limites

PUT /v2/limits

Si le serveur a des limites en base de données, remplace ces limites par celle passées dans la requête.

Example request :

```
PUT /v2/limits HTTP/1.1
Host: example.com
Content-Type: application/json

{
    "business": [
        "server1",
        "server2"
    ],
    "roles": [
        {
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "hostName": "server1",
            "roleList": [
                "TRANSFER",
                "RULE"
            ]
        }
    ],
    "aliases": [
        {
            "hostName": "server1",
            "aliasList": [
                "alias1",
                "alias2"
            ]
        }
    ],
    "others": "<root><version>3.0.12</version></root>"
}
```

Status Codes

— 201 Created – Les limites ont été modifiées avec succès.
Example response :

```
HTTP/1.1 201 Created
Content-Type: application/json

{
    "upGlobalLimit": 1000000000,
    "downGlobalLimit": 50000000,
    "upSessionLimit": 1000000,
    "downSessionLimit": 500000,
    "delayLimit": 1000
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "errorCode": 4
        }
    ]
}
```

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Supprimer les limites

DELETE /v2/limits

Si le serveur a des limites en base de données, supprime ces limites.

Status Codes

— 204 No Content – Les limites ont élé enlevées avec succès.

— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

4.4.8 Commandes serveur

Consulter le statut du serveur

GET /v2/server/status

Renvoie le statut général du serveur sous forme d'un objet JSON regroupant toutes les informations sur celui-ci.

Query Parameters

— **period** (*string*) – La période de temps (en format ISO-8601) sur laquelle le statut du serveur est étudié. (Required)

Example request :

```
GET /v2/server/status?period=string HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "serverName": "string",
    "noSSL": true,
    "SSL": true,
    "date": "2021-08-18T12:47:27.252876",
    "lastRun": "2021-08-18T12:47:27.252876",
    "fromDate": "2021-08-18T12:47:27.252876",
    "secondsRunning": 1,
    "networkConnections": 1,
    "nbThreads": 1,
    "inBandwidth": 1,
    "outBandwidth": 1,
```

(suite sur la page suivante)

(suite de la page précédente)


```
    "overall": {
        "allTransfers": 1,
        "unknown": 1,
        "notUpdated": 1,
        "interrupted": 1,
        "toSubmit": 1,
        "inError": 1,
        "running": 1,
        "done": 1,
        "runningIn": 1,
        "runningOut": 1,
        "lastRunningIn": "2021-08-18T12:47:27.252876",
        "lastRunningOut": "2021-08-18T12:47:27.252876",
        "allIn": 1,
        "allOut": 1,
        "errorsIn": 1,
        "errorsOut": 1
    },
    "globalSteps": {
        "noTask": 1,
        "preTask": 1,
        "transferTask": 1,
        "postTask": 1,
        "allDoneTask": 1,
        "errorTask": 1
    },
    "runningSteps": {
        "allRunning": 1,
        "running": 1,
        "initOk": 1,
        "preProcessingOK": 1,
        "transferOk": 1,
        "postProcessingOk": 1,
        "completeOk": 1
    },
    "errors": {
        "connectionImpossible": 1,
        "serverOverloaded": 1,
        "badAuthent": 1,
        "externalOp": 1,
        "transferError": 1,
        "md5Error": 1,
        "disconnection": 1,
        "remoteShutdown": 1,
        "finalOp": 1,
        "unimplemented": 1,
        "shutdown": 1,
        "remoteError": 1,
        "internal": 1,
        "stopped": 1,
        "canceled": 1,
        "warning": 1,
```


(suite sur la page suivante)

(suite de la page précédente)

```
        "unknown": 1,
        "queryAlreadyFinished": 1,
        "queryStillRunning": 1,
        "unknownHost": 1,
        "remotelyUnknown": 1,
        "fileNotFound": 1,
        "commandNotFound": 1,
        "passThroughMode": 1
    }
}
```

Désactiver le serveur

PUT /v2/server/deactivate

Désactive ou active le serveur pour que celui-ci accepte ou refuse les nouvelles requêtes de transfert.

Status Codes

— 202 Accepted – Le serveur a été (dés)activé avec succès.

Éteindre le serveur

PUT /v2/server/shutdown

Stoppe tous les transferts en cours et arrête le serveur R66.

Status Codes

— 202 Accepted – Le requête a été traitée avec succès, le serveur va s'éteindre.

Redémarrer le serveur

PUT /v2/server/reboot

Stoppe tous les transferts en cours et redémarre le serveur R66.

Status Codes

— 202 Accepted – Le requête a été traitée avec succès, le serveur va redémarrer.

Exporter les logs de transferts

GET /v2/server/logs

Exporte les logs de transferts du serveur vers un fichier XML dans le dossier de logs du serveur.

Query Parameters

— **purge** (*boolean*) – Spécifie si les transferts exportés doivent être purgés de la base de données après l'export de log.
— **clean** (*boolean*) – Spécifie si, avant d'être exportés, les transferts doivent avoir leur statut marqué comme terminé lorsque leur étape globale et sous-étape le permettent.
— **status** (*string*) – Filtre pour n'exporter que les transferts ayant ce statuts.
— **ruleName** (*string*) – Filtre pour n'exporter que les transferts utilisant cette règle de tranfert.
— **start** (*string*) – Filtre pour n'exporter que les transferts ultérieurs à cette date.
— **stop** (*string*) – Filtre pour n'exporter que les transferts antérieurs à cette date.
— **startID** (*integer*) – Filtre pour n'exporter que les transferts ayant un ID plus grand que celui-ci.
— **stopID** (*integer*) – Filtre pour n'exporter que les transferts ayant un ID plus petit que celui-ci.

— **requested** (*string*) – Filtre pour n'exporter que les transferts demandés par cet hôte.

Example request :

```
GET /v2/server/logs HTTP/1.1
Host: example.com
```

Status Codes

— 200 OK – Les logs ont été exportés avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "filePath": "string",
    "exported": 1,
    "purged": 1
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

Importer la configuration

PUT /v2/server/config

Importe divers éléments de la configuration du serveur R66 depuis des fichiers dans le dossier de configuration du serveur.

Query Parameters

— **purgeHosts** (*boolean*) – Spécifie si l'ancienne base de données des hôtes doit être purgée avant d'importer la nouvelle.

— **purgeRules** (*boolean*) – Spécifie si l'ancienne base de données des règles de transfert doit être purgée avant d'importer la nouvelle.

— **purgeBusiness** (*boolean*) – Spécifie si l'ancienne base de données des partenaire de business doit être purgée avant d'importer la nouvelle.
— **purgeAliases** (*boolean*) – Spécifie si l'ancienne base de données des alias doit être purgée avant d'importer la nouvelle.
— **purgeRoles** (*boolean*) – Spécifie si l'ancienne base de données des rôles doit être purgée avant d'importer la nouvelle.
— **hostsFile** (*string*) – Le chemin du fichier contenant les hôtes à importer.
— **rulesFile** (*string*) – Le chemin du fichier contenant les règles à importer.
— **businessFile** (*string*) – Le chemin du fichier contenant les partenaire business à importer.
— **aliasesFile** (*string*) – Le chemin du fichier contenant les alias à importer.
— **rolesFile** (*string*) – Le chemin du fichier contenant les rôles à importer.

Status Codes

— 202 Accepted – La configuration a été importée avec succès.
Example response :

```
HTTP/1.1 202 Accepted
Content-Type: application/json

{
    "purgedHost": true,
    "purgedRule": true,
    "purgedBusiness": true,
    "purgedAlias": true,
    "purgedRoles": true,
    "importedHost": true,
    "importedRule": true,
    "importedBusiness": true,
    "importedAlias": true,
    "importedRoles": true
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

Exporter la configuration

GET /v2/server/config

Exporte divers éléments de la configuration du serveur R66 vers des fichiers dans le dossier de configuration du serveur.

Query Parameters

- **exportHosts** (*boolean*) – Spécifie si la liste des hôtes connus du serveur doit être exportée non.
- **exportRules** (*boolean*) – Spécifie si les règles de transfert doivent être exportées ou non.
- **exportBusiness** (*boolean*) – Spécifie si la liste des hôtes autorisés à exécuter un business sur le serveur doit être exportée ou non.
- **exportAliases** (*boolean*) – Spécifie si la liste des alias connus de chaque hôte doit être exportée ou non.
- **exportRoles** (*boolean*) – Spécifie si la liste des rôles autorisés de chaque hôte doit être exportée ou non.

Example request :

```
GET /v2/server/config HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – La configuration a été exportée avec succès.
  Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "fileHost": "string",
    "fileRule": "string",
    "fileBusiness": "string",
    "fileAlias": "string",
    "fileRoles": "string"
}
```

- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
  Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
```

(suite sur la page suivante)

(suite de la page précédente)

```
            "errorCode": 4
        }
    ]
}
```

4.4.9 Gestion des FileMonitors

Lister les FileMonitors

Nouveau dans la version 3.5.1 : Ajout du service

GET /v2/filemonitors

Renvoie une liste des FileMonitors reçus par le serveur respectant les filtres donnés en paramètres de requête, ou simplement le décompte si countOrder=true est passé en paramètre.

Query Parameters

- **name** (*string*) – Nom optionnel du FileMonitor à retourner
- **status** (*integer*) – 0 pour tous les FileMonitors, 1 pour ceux actifs, -1 pour les inactifs.
- **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Example request :

```
GET /v2/filemonitors HTTP/1.1
Host: example.com
```

Status Codes

- 200 OK – La requête s'est déroulée avec succès.

Example response :

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "totalResults": 1,
    "totalSubResults": 1,
    "results": [
        {
            "NAME": "string",
            "HOST": "string",
            "LAST_UPDATE": 1,
            "GLOBALOK": 1,
            "GLOBALERROR": 1,
            "TODAYOK": 1,
            "TODAYERROR": 1,
            "INTERVAL": 1,
            "STOPFILE": "string",
            "STATUSFILE": "string",
            "SUBDIRS": true,
            "DIRECTORIES": "string",
            "FILES": {}
        }
```

(suite sur la page suivante)

(suite de la page précédente)

```
    ]
}
```

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
Example response :

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "errors": [
        {
            "message": "Query parameter 'startTrans' : illegal value
↪'not_a_date'.",
            "errorCode": 4
        },
        {
            "message": "Query parameter 'status' : illegal value 'not_
↪a_status'.",
            "errorCode": 4
        }
    ]
}
```

4.4.10 Documentation complète

GET /v2/transfers

Renvoie une liste des transferts monitorés par le serveur respectant les filtres donnés en paramètres de requête, ou simplement le décompte si countOrder=true est passé en paramètre.

Query Parameters

— **limit** (*integer*) – Nombre maximal de transferts pouvant être inclus dans la réponse.
— **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse.
— **order** (*string*) – Ordre dans lequel les réponses seront triées. Valeurs possibles : *ascId*, *descId*, *ascFile*, *descFile*, *ascStart*, *descStart*, *ascStop*, *descStop*
— **ruleName** (*string*) – Filter les transferts par règle de transfert.
— **partner** (*string*) – Filtrer les transferts par partenaire.
— **status** (*string*) – Filtrer les transferts par statut. Valeurs possibles : *TOSUBMIT*, *NOTUPDATED*, *RUNNING*, *INTERRUPTED*, *DONE*, *INERROR*, *UNKNOWN*
— **filename** (*string*) – Filtrer les transferts par fichier.
— **startTrans** (*string*) – Filtrer les transferts commençant après cette date (format ISO-8601).
— **stopTrans** (*string*) – Filtrer les transferts commençant avant cette date (format ISO-8601).
— **followId** (*string*) – Identifiant FollowId à rechercher dans les transferts.
— **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Status Codes

— 200 OK – La requête s'est déroulée avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.

Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

`POST /v2/transfers`

Initialise un nouveau transfert sur le serveur avec les attributs définis dans le corps de la requête. Les informations sur le transfert créé sont renvoyées dans la réponse ainsi que l'URI pour le consuter.

Status Codes

- 201 Created – Le nouveau transfert a été créé avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Response Headers

- **transferURI** – L'URI d'accès pour consulter le transfert.

`GET /v2/transfers/{id}`

Si le transfert demandé existe, renvoie toutes ses informations.

Parameters

- **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

- 200 OK – La requête s'est déroulée avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`PUT /v2/transfers/{id}/restart`

S'il existe, redémarre le transfert demandé.

Parameters

- **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

- 202 Accepted – Le transfert a été redémarré avec succès.

`PUT /v2/transfers/{id}/stop`

S'il existe, arrête le transfert demandé.

Parameters

- **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

- 202 Accepted – Le transfert a été arrêté avec succès.

`PUT /v2/transfers/{id}/cancel`

If it exists, cancels the requested transfer.

Parameters

- **id** (*string*) – L'identifiant unique du transfert. Afin de garantir l'unicité de cet id dans le cas d'une base de donnée partagée, ce paramètre est en fait composé de l'id du transfert sur le serveur, ainsi que du nom de l'hôte destinataire de ce transfert (i.e. le champ *requested* du transfert).

Status Codes

— 202 Accepted – Le transfert a été annulé avec succès.

`GET /v2/hosts`

Renvoie l'ensemble des hôtes conformes aux filtres passés en paramètres de requête. Si le paramètre countOrder=true est donnée, seule le décompte est retournée. Les hôtes peuvent être filtrés par addresse, sur leur utilisation de SSL, ou sur le fait qu'ils soient actifs ou non. Les paramètres *offset* et *limit* permettent de fixer un numéro de départ et un nombre maximal d'hôtes désirés, afin de ne renvoyer qu'un sous-ensemble de la liste demandée. Le paramètre *order* spécifie dans quel ordre les hôtes doivent être ordonnées.

Query Parameters

— **limit** (*integer*) – Fixe le nombre maximal de transferts pouvant être inclus dans la réponse. Utile lorsque le nombre d'entrées sélectionnées par les filtres est trop important pour tenir dans un seule message.

— **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse. Généralement utilisé en combinaison avec le paramètre *limit* pour restreindre l'ensemble des hôtes à renvoyer. **Attention** : Si le *offset* est plus grand que le nombre total de réponses, aucune entrée ne sera renvoyée.

— **order** (*string*) – Spécifie l'attribut à utiliser pour trier les éléments de la réponse, ainsi que le sens du tri. Ce paramètre devrait donc être le nom d'un attribut d'hôte préfixé par « asc » ou « desc », correspondant respectivement à l'ordre croissant et décroissant.

— **address** (*string*) – Si ce paramètre est défini, la réponse ne contiendra que les hôtes ayant cette adresse.

— **isSSL** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que les hôtes autorisant les transferts SSL. Et si le paramètre est faux, que ceux ne les autorisant pas. Si le paramètre n'est pas défini, aucun filtrage ne sera fait.

— **isActive** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que les hôtes actifs. Et si le paramètre est faux, que ceux inactifs. Si le paramètre n'est pas défini, aucun filtrage ne sera fait.

— **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

— 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

`POST /v2/hosts`

Ajoute un nouvel hôte à la base de données avec les attribut de l'objet passé dans le corps de la requête. La nouvelle entrée créée est renvoyée dans la réponse, ainsi que l'URI pour la consulter.

Status Codes

— 201 Created – Le nouvel hôte a été ajouté avec succès.

— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

— 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

Response Headers

— **hostURI** – L'URI d'accès aux informations du nouvel hôte.

`GET /v2/hosts/{id}`
S'il existe, renvoie les informations de l'hôte demandé sous forme d'un objet JSON.

Parameters
- **id** (*string*) – Le nom unique de l'hôte désiré.

Status Codes
- 200 OK – L'hôte demandé a été trouvé.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`PUT /v2/hosts/{id}`
S'il existe, remplace les informations de l'hôte demandé par celles données dans la requête. Les nouvelles informations seront renvoyées dans la réponse, ainsi que leur URI d'accès.

Parameters
- **id** (*string*) – Le nom unique de l'hôte désiré.

Status Codes
- 202 Accepted – L'hôte a été mis à jour avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.
- 415 Unsupported Media Type – Le type de contenu de la requête n'est pas valide pour cette requête.

Response Headers
- **host-uri** – Le chemin d'accès de l'hôte modifié. Si le nom d'hôte n'a pas été changé, cet URI sera identique à celui ayant été utilisé pour la requête.

`DELETE /v2/hosts/{id}`
S'il existe, supprime l'hôte de la base de données.

Parameters
- **id** (*string*) – Le nom unique de l'hôte désiré.

Status Codes
- 204 No Content – L'hôte a été supprimé avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`GET /v2/hostconfig`
Renvoie la configuration du serveur sous forme d'un objet JSON.

Status Codes
- 200 OK – La requête s'est déroulée avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`POST /v2/hostconfig`
Initialise la configuration du serveur dans la base de données avec les attribut passés dans la requête.

Status Codes
- 201 Created – La configuration du serveur a été ajoutée avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`PUT /v2/hostconfig`
Si le serveur a une configuration en base de données, remplace cette configuration avec les nouveaux attributs passé dans la requête.

Status Codes
- 202 Accepted – La configuration a été mise à jour avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`DELETE /v2/hostconfig`
Si le serveur a une configuration en base de données, supprime cette configuration.

Status Codes
- 204 No Content – La configuration a été supprimée avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`GET /v2/limits`
Renvoie les limites de bande passante sous forme d'un objet JSON.

Status Codes
- 200 OK – La requête s'est déroulée avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`POST /v2/limits`
Initialise les limites de bande passante du serveur avec les attributs passés dans la requête.

Status Codes
- 201 Created – Les limites ont été initialisées avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`PUT /v2/limits`
Si le serveur a des limites en base de données, remplace ces limites par celle passées dans la requête.

Status Codes
- 201 Created – Les limites ont été modifiées avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`DELETE /v2/limits`
Si le serveur a des limites en base de données, supprime ces limites.

Status Codes
- 204 No Content – Les limites ont élé enlevées avec succès.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`GET /v2/rules`

Renvoie l'ensemble des règles de transfert conformes aux filtres passés en paramètres de requête. Si le paramètre countOrder=true est donnée, seule le décompte est retournée. Les règles peuvent être filtrées par mode de transmission. Les paramètres *offset* et *limit* permettent de fixer un numéro de départ et un nombre maximal de règles désirées, afin de ne renvoyer qu'un sous-ensemble de la liste demandée. Le paramètre *order* spécifie dans quel ordre les règles doivent être ordonées.

Query Parameters

- **limit** (*integer*) – Fixe le nombre maximal de règles pouvant être inclues dans la réponse. Utile lorsque le nombre d'entrées sélectionnées par les filtres est trop important pour tenir dans un seule message.
- **offset** (*integer*) – Indice de la première entrée à inclure dans la réponse. Généralement utilisé en combinaison avec le paramètre *limit* pour restreindre l'ensemble des hôtes à renvoyer. **Attention** : Si le *offset* est plus grand que le nombre total de réponses, aucune entrée ne sera renvoyée.
- **order** (*string*) – Spécifie l'attribut à utiliser pour trier les éléments de la réponse, ainsi que le sens du tri. Ce paramètre devrait donc être le nom d'un attribut d'hôte préfixé par « asc » ou « desc », correspondant respectivement à l'ordre croissant et décroissant.
- **modeTrans** (*string*) – Si ce paramètre est défini, la réponse ne contiendra que les règles utilisant ce mode de transfert.
- **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Status Codes

- 200 OK – La requête s'est déroulée avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

`POST /v2/rules`

Creates a new transfer rule with the ObjectRule passed in the request body. The created entry is returned in the response body, along with its URI.

Status Codes

- 201 Created – La nouvelle règle de transfert a été ajoutée avec succès.
- 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
  Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

Response Headers

- **ruleURI** – L'URI d'accès à la nouvelle règle créée.

`GET /v2/rules/{id}`

Si elle existe, renvoie les informations de la règle demandée sous forme d'un objet JSON.

Parameters

- **id** (*string*) – L'identifiant unique de la règle souhaitée.

Status Codes

- 200 OK – La règle remandée a été trouvée.
- 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`PUT /v2/rules/{id}`

Si elle existe, remplace les informations de la règle demandée par celles données dans la requête. Les nouvelles informations seront renvoyées dans la réponse, ainsi que leur URI d'accès.

Parameters

— **id** (*string*) – L'identifiant unique de la règle souhaitée.

Status Codes

— 201 Created – La règle de transfert a été mise à jour avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.
— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

Response Headers

— **ruleURI** – Le chemin d'accès de la règle modifiée. Si le nom de la règle n'a pas été changé, cet URI sera identique à celui ayant été utilisé pour la requête.

`DELETE /v2/rules/{id}`

Si elle existe, supprime la règle de transfert de la base de données.

Parameters

— **id** (*string*) – L'identifiant unique de la règle souhaitée.

Status Codes

— 204 No Content – La règle a été supprimée avec succès.
— 404 Not Found – Dans le cas d'une requête sur une collection, cela signifie que l'identifiant donné n'existe pas dans la base de données.

`GET /v2/server/status`

Renvoie le statut général du serveur sous forme d'un objet JSON regroupant toutes les informations sur celui-ci.

Query Parameters

— **period** (*string*) – La période de temps (en format ISO-8601) sur laquelle le statut du serveur est étudié. (Required)

Status Codes

— 200 OK – La requête s'est déroulée avec succès.

`PUT /v2/server/deactivate`

Désactive ou active le serveur pour que celui-ci accepte ou refuse les nouvelles requêtes de transfert.

Status Codes

— 202 Accepted – Le serveur a été (dés)activé avec succès.

`PUT /v2/server/shutdown`

Stoppe tous les transferts en cours et arrête le serveur R66.

Status Codes

— 202 Accepted – Le requête a été traitée avec succès, le serveur va s'éteindre.

`PUT /v2/server/reboot`

Stoppe tous les transferts en cours et redémarre le serveur R66.

Status Codes

— 202 Accepted – Le requête a été traitée avec succès, le serveur va redémarrer.

`GET /v2/server/logs`

Exporte les logs de transferts du serveur vers un fichier XML dans le dossier de logs du serveur.

Query Parameters

— **purge** (*boolean*) – Spécifie si les transferts exportés doivent être purgés de la base de données après l'export de log.
— **clean** (*boolean*) – Spécifie si, avant d'être exportés, les transferts doivent avoir leur statut marqué comme terminé lorsque leur étape globale et sous-étape le permettent.
— **status** (*string*) – Filtre pour n'exporter que les transferts ayant ce statuts.
— **ruleName** (*string*) – Filtre pour n'exporter que les transferts utilisant cette règle de tranfert.

— **start** (*string*) – Filtre pour n'exporter que les transferts ultérieurs à cette date.
— **stop** (*string*) – Filtre pour n'exporter que les transferts antérieurs à cette date.
— **startID** (*integer*) – Filtre pour n'exporter que les transferts ayant un ID plus grand que celui-ci.
— **stopID** (*integer*) – Filtre pour n'exporter que les transferts ayant un ID plus petit que celui-ci.
— **requested** (*string*) – Filtre pour n'exporter que les transferts demandés par cet hôte.

Status Codes
— 200 OK – Les logs ont été exportés avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

`GET /v2/server/config`
Exporte divers éléments de la configuration du serveur R66 vers des fichiers dans le dossier de configuration du serveur.

Query Parameters
— **exportHosts** (*boolean*) – Spécifie si la liste des hôtes connus du serveur doit être exportée non.
— **exportRules** (*boolean*) – Spécifie si les règles de transfert doivent être exportées ou non.
— **exportBusiness** (*boolean*) – Spécifie si la liste des hôtes autorisés à exécuter un business sur le serveur doit être exportée ou non.
— **exportAliases** (*boolean*) – Spécifie si la liste des alias connus de chaque hôte doit être exportée ou non.
— **exportRoles** (*boolean*) – Spécifie si la liste des rôles autorisés de chaque hôte doit être exportée ou non.

Status Codes
— 200 OK – La configuration a été exportée avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

`PUT /v2/server/config`
Importe divers éléments de la configuration du serveur R66 depuis des fichiers dans le dossier de configuration du serveur.

Query Parameters
— **purgeHosts** (*boolean*) – Spécifie si l'ancienne base de données des hôtes doit être purgée avant d'importer la nouvelle.
— **purgeRules** (*boolean*) – Spécifie si l'ancienne base de données des règles de transfert doit être purgée avant d'importer la nouvelle.
— **purgeBusiness** (*boolean*) – Spécifie si l'ancienne base de données des partenaire de business doit être purgée avant d'importer la nouvelle.
— **purgeAliases** (*boolean*) – Spécifie si l'ancienne base de données des alias doit être purgée avant d'importer la nouvelle.
— **purgeRoles** (*boolean*) – Spécifie si l'ancienne base de données des rôles doit être purgée avant d'importer la nouvelle.
— **hostsFile** (*string*) – Le chemin du fichier contenant les hôtes à importer.
— **rulesFile** (*string*) – Le chemin du fichier contenant les règles à importer.
— **businessFile** (*string*) – Le chemin du fichier contenant les partenaire business à importer.
— **aliasesFile** (*string*) – Le chemin du fichier contenant les alias à importer.
— **rolesFile** (*string*) – Le chemin du fichier contenant les rôles à importer.

Status Codes

— 202 Accepted – La configuration a été importée avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

`GET /v2/filemonitors`

Renvoie une liste des FileMonitors reçus par le serveur respectant les filtres donnés en paramètres de requête. Si le paramètre countOrder=true est donnée, seule le décompte est retournée.

Query Parameters

— **name** (*string*) – Nom optionnel du FileMonitor à retourner
— **status** (*integer*) – 0 pour tous les FileMonitors, 1 pour ceux actifs, -1 pour les inactifs.
— **countOrder** (*boolean*) – Si ce paramètre est vrai, la réponse ne contiendra que le nombre des hôtes selon les conditions, sinon la liste réelle. Si le paramètre n'est pas défini, la liste est retournée.

Status Codes

— 200 OK – La requête s'est déroulée avec succès.
— 400 Bad Request – La requête est invalide, soit à cause de paramètres de requête erronés, soit à cause d'un corps de requête invalide.
Les causes de l'échec de la requête sont données dans le corps de la réponse sour forme d'un tableau JSON.

CHAPITRE 5

Référence des commandes

5.1 Waarp R66 Server

Le script `waarp-r66server` (ou `./bin/waarp-r66server.sh` dans les archives portables) exécutent des appels `java` qui utilisent les commandes fournies par Waarp R66.

L'utilisation des scripts est préférée pour les raisons suivantes :
- Ils construisent automatiquement les commandes java, qui sont laborieuses à saisir ;
- Ils fournissent une indirection pour gérer les installations multi-instances
- Ils sont plus ergonomiques.

Le détail des commandes java brutes est disponible *ici*.

Sommaire

5.1.1 Gestion du service

Le serveur Waarp R66 peut être démarré et arrêté avec le script `waarp-r66server`.

Les commandes suivantes sont disponibles :

Commande `waarp-r66server start`

Démarrage du serveur.

Codes de retour :

Code	Signification
0	Succès
1	Erreur au démarrage ou le serveur est déjà démarré

Commande `waarp-r66server stop`

Arrêt du serveur

Codes de retour :

Code	Signification
0	Succès
1	Le serveur est déjà arrêté ou le signal d'arrêt n'a pas pu être envoyé au serveur

Commande `waarp-r66server status`

Statut du serveur (démarré ou arrêté)

Codes de retour :

Code	Signification
0	Le serveur est démarré
1	le serveur est arrêté

Commande `waarp-r66server restart`

Redémarrage du serveur

Codes de retour :

Code	Signification
0	Succès
1	Erreur au démarrage ou le serveur est déjà démarré

5.1.2 Autres commandes

D'autres commandes de maintenance sont disponibles pour le même exécutable `waarp-r66server`.

Commande `waarp-r66server initdb`

Initialise la base de données.

Syntaxe d'appel :

```
waarp-r66server initdb [OPTIONS]
```

Cette commande accepte les arguments suivants :

-initdb
: Initialise la base de données

-upgradeDb
: Met à jour le modèle de la base de données

-dir DOSSIER
: Charge les règles de transferts en base depuis le dossier DOSSIER

-auth FICHIER
: Charge les données d'authentification en base depuis le fichier FICHIER

-limit FICHIER
: Charge les limitation de bande passante en base depuis le fichier FICHIER

-loadAlias FICHIER
: Charge les alias du serveur en base en base depuis le fichier FICHIER

-loadRoles FICHIER
: Charge les rôles du serveur en base depuis le fichier FICHIER

-loadBusiness FICHIER
: Charge les données business en base depuis le fichier FICHIER

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
2	Une erreur SQL s'est produite durant l'initialisation de la base

Commande `waarp-r66server loadauth`

Charge les données d'authentification depuis un fichier XML donné en argument dans la base de données de WaarpR66 Server.

Syntaxe d'appel :

```
waarp-r66server loadauth /path/to/authent.xml
```

Codes de retour :

Code	Signification
0	Succès
1	WaarpR66 a retourné une erreur durant le chargement des données en base.

Commande `waarp-r66server loadrule`

Charge les règles de transfert depuis un dossier donné en argument dans la base de données de WaarpR66 Server.

Syntaxe d'appel :

```
waarp-r66server loadrule /path/to/rules_dir
```

Codes de retour :

Code	Signification
0	Succès
1	WaarpR66 a retourné une erreur durant le chargement des données en base.

Commande `waarp-r66server loadconf`

Charge la configuration (authentification et règles de transfert) depuis des fichiers XML dans la base de données de WaarpR66 Server. Il s'agit d'un raccourci vers les deux commandes `loadauth` et `loadrule`.

Les fichiers attendus par la commande sont les suivants :
- `/etc/waarp/{hostid}/authent-server.xml` : fichier contenant les données d'authentification
- `/etc/waarp/{hostid}/` : dossier contenant les définitions de règles

Une fois les données chargées en base de données, les fichiers peuvent être supprimés sans risque.

Codes de retour :

Code	Signification
0	Succès
1	WaarpR66 a retourné une erreur durant le chargement des données en base.

5.2 Waarp R66 Client

Le script `waarp-r66client` (ou `./bin/waarp-r66client.sh` dans les archives portables) exécutent des appels `java` qui utilisent les commandes fournies par Waarp R66.

L'utilisation des scripts est préférée pour les raisons suivantes :
- Ils construisent automatiquement les commandes java, qui sont laborieuses à saisir ;
- Ils fournissent une indirection pour gérer les installations multi-instances
- Ils sont plus ergonomiques.

Le détail des commandes java brutes est disponible *ici*.

Sommaire

5.2.1 Commandes de gestion des transferts

Commande `waarp-r66client send`

Démarre un transfert synchrone (attend le résultat du transfert avant de rendre la main).

Cette commande accepte les arguments suivants :

`-to` `PARTNER`
: *obligatoire*
Serveur R66 de destination

`-file` `FILENAME`
: *obligatoire pour démarrer un nouveau transfert*
Fichier à envoyer

`-rule` `RULE`
: *obligatoire pour démarrer un nouveau transfert*
Règle de transfert à utiliser

`-id`
: *obligatoire pour relancer un transfert*
Identifiant du transfert à relancer

`-info` `INFO`
: Info complémentaires sur le transfert

`-block`
: Fixe la taille de blocs pour le transfert

`-md5`
: Force le contrôle d'intégrité par paquet (déconseillé)

`-nolog`
: Désactive les logs pour ce transfert

`-logWarn`
: Loggue les messages INFO avec un niveau WARN

`-notlogWarn`
: Loggue les messages INFO avec un niveau INFO

-nofollow
Nouveau dans la version 3.4.0.
Désactive l'assignation d'un identifiant de suivi de transfert pour ce fichier

Codes de retour :

Code	Signification
0	Succès
2	Les arguments sont incorrects ou le fichier de configuration contient une erreur
66	Une erreur inattendue s'est produite
N	Les autres codes de sortie correspondent à une erreur de transfert. Il s'agit de la valeur numérique du *code d'erreur*

Commande `waarp-r66client asend`

Démarre un transfert asynchrone (enregistre le démarrage du transfert et de rendre la main).

Cette commande accepte les arguments suivants :

-to PARTNER
obligatoire
Serveur R66 de destination

-file FILENAME
obligatoire pour démarrer un nouveau transfert
Fichier à envoyer

-rule RULE
obligatoire pour démarrer un nouveau transfert
Règle de transfert à utiliser

-id
obligatoire pour relancer un transfert
Identifiant du transfert à relancer

-info INFO
Info complémentaires sur le transfert

-block
Fixe la taille de blocs pour le transfert

-md5
Force le contrôle d'intégrité par paquet (déconseillé)

-nolog
Désactive les logs pour ce transfert

-logWarn
Loggue les messages INFO avec un niveau WARN

-notlogWarn
Loggue les messages INFO avec un niveau INFO

-start yyyyMMddHHmmss
Date à laquelle le transfert doit démarrer

-delay timestamp|+NNN
Si un timestamp est fourni, date à laquelle le transfert doit démarrer (sous la forme d'un timestamp UNIX en ms).

Si une valeur de la forme +NNN est fournie, délais en seconde à partir de l'exécution de la commande après lequel le transfert doit démarrer

-nofollow
Nouveau dans la version 3.4.0.
Désactive l'assignation d'un identifiant de suivi de transfert pour ce fichier

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
N	Les autres codes de sortie correspondent à une erreur de transfert. Il s'agit de la valeur numérique du *code d'erreur*

Commande waarp-r66client msend

Démarre plusieurs transferts synchrones (attend le résultat du transfert avant de rendre la main).

Cette commande fonctionne sensiblement comme la commande send, mais permet de lister plusieurs fichiers et plusieurs hôtes de destination :
- En séparant les valeurs dans les arguments -to et -file par des virgules (,)
- En utilisant des « jokers » dans l'argument file (* pour remplacer plusieurs caractères ou ? pour remplacer un caractère unique.)

Cette commande accepte les arguments suivants :

-to PARTNER
obligatoire
Serveur R66 de destination

-file FILENAME
obligatoire pour démarrer un nouveau transfert
Fichier à envoyer

-rule RULE
obligatoire pour démarrer un nouveau transfert
Règle de transfert à utiliser

-id
obligatoire pour relancer un transfert
Identifiant du transfert à relancer

-info INFO
Info complémentaires sur le transfert

-block
Fixe la taille de blocs pour le transfert

-md5
Force le contrôle d'intégrité par paquet (déconseillé)

-nolog
Désactive les logs pour ce transfert

-logWarn
Loggue les messages INFO avec un niveau WARN

-notlogWarn
Loggue les messages INFO avec un niveau INFO

-nofollow
Nouveau dans la version 3.4.0.
Désactive l'assignation d'un identifiant de suivi de transfert pour ce fichier

Codes de retour :

Code	Signification
0	Succès
2	Les arguments sont incorrects ou le fichier de configuration contient une erreur
22	Erreur inconnue
N	Nombre de transferts en erreur

Commande `waarp-r66client masend`

Démarre plusieurs transferts asynchrones (enregistre le démarrage du transfert et de rendre la main).

Cette commande fonctionne sensiblement comme la commande `asend`, mais permet de lister plusieurs fichiers et plusieurs hôtes de destination :

— En séparant les valeurs deans les arguments `-to` et `-file` par des virgules (",")
— En utilisant des « jokers » dans l'argument `file` (`*` pour remplacer plusieurs caractères ou `?` pour remplacer un caractère unique.)

Cette commande accepte les arguments suivants :

-to PARTNER
obligatoire
Serveur R66 de destination

-file FILENAME
obligatoire pour démarrer un nouveau transfert
Fichier à envoyer

-rule RULE
obligatoire pour démarrer un nouveau transfert
Règle de transfert à utiliser

-id
obligatoire pour relancer un transfert
Identifiant du transfert à relancer

-client
Doit être ajouté pour si la règle est en mode réception

-info INFO
Info complémentaires sur le transfert

-block
Fixe la taille de blocs pour le transfert

-md5
Force le contrôle d'intégrité par paquet (déconseillé)

-nolog
Désactive les logs pour ce transfert

-logWarn
Loggue les messages INFO avec un niveau WARN

-notlogWarn
Loggue les messages INFO avec un niveau INFO

-start yyyyMMddHHmmss
: Date à laquelle le transfert doit démarrer

-delay timestamp|+NNN
: Si un timestamp est fourni, date à laquelle le transfert doit démarrer (sous la forme d'un timestamp UNIX en ms).

Si une valeur de la forme +NNN est fournie, délais en seconde à partir de l'exécution de la commande après lequel le transfert doit démarrer

-nofollow
: Nouveau dans la version 3.4.0.

Désactive l'assignation d'un identifiant de suivi de transfert pour ce fichier

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
2	Erreur de connexion à la base de données ou absence de l'argument -client
N	Nombre de transferts dont la programmation est en erreur

Commande waarp-r66client transfer

Cette commande permet d'obtenir des informations sur un transfert en cours ou terminé, et d'agir sur ces transferts

Elle accepte les arguments suivants :

-id
: *obligatoire*

Identifiant du transfert

-to
: *Les options -to et -from sont exclusives, et l'une des deux doit être fournie*

Partenaire de destination

-from
: *Les options -to et -from sont exclusives, et l'une des deux doit être fournie*

Partenaire de d'origine

-cancel
: *Les options -cancel, -stop et -restart sont exclusives*

Annule le transfert en cours (les fichiers temporaires sont supprimés sur le récepteur)

-stop
: *Les options -cancel, -stop et -restart sont exclusives*

Arrête un transfert en cours

-restart
: *Les options -cancel, -stop et -restart sont exclusives*

Redémarre un transfert en erreur

-start yyyyMMddHHmmss
: *Ne peut être utilisé qu'avec l'action -restart*

Date à laquelle le transfert doit démarrer

-delay timestamp|+NNN
: Si un timestamp est fourni, date à laquelle le transfert doit démarrer (sous la forme d'un timestamp UNIX en ms).

Si une valeur de la forme +NNN est fournie, délais en seconde à partir de l'exécution de la commande après lequel le transfert doit démarrer

Codes de retour communs :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
99	Une erreur inattendue s'est produite

Codes de retour pour l'action *-cancel* :

Code	Signification
3	Le transfert est déjà terminé
4	L'action demandée n'a pas pu être effectuée

Codes de retour pour l'action *-stop* :

Code	Signification
3	L'action demandée n'a pas pu être effectuée

Codes de retour pour l'action *-restart* :

Code	Signification
3	L'action demandée n'a pas pu être effectuée
4	Le transfert est déjà terminé
5	Le partenaire distant a renvoyé une erreur

Commande `waarp-r66client getinfo`

Cette commande permet d'obtenir sur les fichiers disponibles sur un partenaire distant.

Elle accepte les arguments suivants :

-to PARTNER
Obligatoire
Serveur R66 de destination

-file FILENAME
Obligatoire
Fichier à envoyer (peut contenir des caractères de subtitution « * »)

-rule RULE
Règle de transfert à utiliser

-exist
Vérifie si le fichier donné exist

-detail
Récupère des informations sur le fichie

-list
Liste les fichiers correspondant au motif donn

-mlsx
Liste les fichiers et récupère leurs détails

Codes de retour communs :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
2	Une erreur s'est produite durant l'interrogation du partenaire
3	Une erreur inattendue s'est produite

Commande `waarp-r66client gui`

Ouvre un client graphique pour démarrer un transfert.

Avertissement : Ne fonctionne que dans un environnement graphique

5.2.2 Autres commandes

Commande `waarp-r66client initdb`

Initialise la base de données du client.

Cette commande accepte les arguments suivants :

-initdb
Initialise la base de données

-upgradeDb
Met à jour le modèle de la base de données

-dir DOSSIER
Charge les règles de transferts en base depuis le dossier DOSSIER

-auth FICHIER
Charge les données d'authentification en base depuis le fichier FICHIER

-limit FICHIER
Charge les limitation de bande passante en base depuis le fichier FICHIER

-loadAlias FICHIER
Charge les alias du serveur en base en base depuis le fichier FICHIER

-loadRoles FICHIER
Charge les rôles du serveur en base depuis le fichier FICHIER

-loadBusiness FICHIER
Charge les données business en base depuis le fichier FICHIER

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
2	Une erreur SQL s'est produite durant l'initialisation de la base

Commande `waarp-r66client loadconf`

Charge la configuration (authentification et règles de transfert) depuis des fichiers XML dans la base de données de WaarpR66 Server. Il s'agit d'un raccourci vers les deux commandes `loadauth` et `loadrule`.

Les fichiers attendus par la commande sont les suivants :
- `/comp/waarp/wrs/etc/authent-server.xml` : fichier contenant les données d'authentification
- `/comp/waarp/wrs/etc/` : dossier contenant les définitions de règles

Une fois les données chargées en base de données, les fichiers peuvent être supprimés sans risque.

Codes de retour :

Code	Signification
`0`	Succès
1	WaarpR66 a retourné une erreur durant le chargement des données en base.

Commande `waarp-r66client log-export`

Cette commande permet d'exporter l'historique de transfert du serveur WaarpR66 associé au client, et le cas échéant de purger l'historique.

Les fichiers XML produit sont déposés dans le dossier `arch` définitions dans la configuration du serveur.

> **Avertissement :** Cette commande ne fonctionne que pour les clients associés à un serveur WaarpR66.
>
> Elle sera déplacée dans le script waarp-r66server.sh dans une version future

Cette commande accepte les arguments suivants :

`-clean`
: Corrige le statut des transferts terminés erronés

`-purge`
: Supprime l'historique exporté de la base de données

`-start` `DATE`
: Exporte seulement l'historique postérieur à cette date

`-stop` `DATE`
: Exporte seulement l'historique antérieur à cette date

`-startid` `ID`
: Valeur minimale d'identifiants de transfert à exporter

`-stopid` `ID`
: Valeur maximale d'identifiants de transfert à exporter

`-rule` `RULE`
: Limite l'export à une règle spécifique

`-request` `HOST`
: Limite l'export à un partenaire spécifique

`-pending`
: Limite l'export aux transferts en attente

`-transfer`
: Limite l'export aux transferts en cours

-done
Limite l'export aux transferts terminés

-error
Limite l'export aux transferts en erreur

Les valeurs `DATE` doivent avoir le format `yyyyMMddHHmmssSSS`. La date peut omettre les derniers éléments (ex : `20150815`).

Codes de retour :

Code	Signification
`0`	Succès
`1`	Les arguments sont incorrects ou le fichier de configuration contient une erreur
`10`	Le serveur WaarpR66 associé au client n'est pas trouvé
`20`	Warning
`N`	Les autres codes de sortie correspondent à une erreur de transfert. Il s'agit de la valeur numérique du *code d'erreur*

Commande `waarp-r66client config-export`

Cette commande permet d'exporter la configuration enregistrée en base de données du serveur WaarpR66 associé au client.

Les fichiers XML produit sont déposés dans le dossier `arch` définitions dans la configuration du serveur.

> **Avertissement :** Cette commande ne fonctionne que pour les clients associés à un serveur WaarpR66.
>
> Elle sera déplacée dans le script waarp-r66server.sh dans une version future

Cette commande accepte les arguments suivants :

-hosts
Exporte les données d'authentification

-rules
Exporte les règles de transfert

-business
Exporte les données business

-alias
Exporte les alias du serveur

-role
Exporte les rôles du serveur

-host `HOST`
Envoi la demande d'export au serveur HOST

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
10	Le serveur WaarpR66 associé au client n'est pas trouvé
20	Warning
N	Les autres codes de sortie correspondent à une erreur de transfert. Il s'agit de la valeur numérique du *code d'erreur*

Commande `waarp-r66client icaptest`

Nouveau dans la version 3.4.0.

Cette commande permet de tester les arguments d'une tâche ICAP en envoyant un fichier local à un serveur ICAP.

Cette commande accepte les arguments suivants :

`-file` `FILENAME`

Obligatoire

Spécifie le chemin du fichier à envoyer au serveur ICAP.
Si la valeur donnée est `EICARTEST`, un faux virus basé sur le test EICAR sera envoyé).

`-to` `HOST`

Obligatoire

L'adresse du serveur ICAP.

`-port` `PORT`

Le port du serveur ICAP.

`-service` `SERVICE`

Au moins l'un des arguments ``-service`` ou ``-model`` doit être donné

Le nom du service à utiliser sur le serveur distant.

`-model` `MODEL`

Au moins l'un des arguments ``-service`` ou ``-model`` doit être donné

Le Modèle de service à utiliser pour le serveur distant.

`-maxSize` `SIZE`

Défaut 2147483647

La taille maximale du fichier à envoyer. Si le fichier donné à l'option `-file` a une taille supérieure à cette valeur, rien ne sera transmis au serveur.

`-previewSize` `SIZE`

Défaut Négocié avec le serveur

La taille de *preview* à utiliser.

`-blockSize` `SIZE`

Défaut 8192

Spécifie la taille de bloc utilisée.

`-receiveSize` `SIZE`

Défaut 65536

Spécifie la taille à recevoir

`-timeout` `DURATION`

Délais d'attente de réponse (en ms).

-keyPreview
Spécifie la clef à chercher dans la réponse d'un *preview* pour valider le fichier.

-stringPreview
Spécifie la valeur associée à la clef `-keyPreview` à chercher dans la réponse pour valider le fichier.

-key204
Spécifie la clef à chercher dans une réponse 204 du serveur pour valider le fichier.

-string204
Spécifie la valeur associée à la clef `-key204` à chercher dans une réponse 204 du serveur pour valider le fichier.

-key200
Spécifie la clef à chercher dans une réponse 200 du serveur pour valider le fichier.

-string200
Spécifie la valeur associée à la clef `-key200` à chercher dans une réponse 200 du serveur pour valider le fichier.

-stringHttp
Spécifie une valeur à rechercher dans le statut d'une réponse 200 du serveur pour valider le fichier.

-errorDelete
Supprime les fichiers considérés invalides

-errorMove `PATH`
Déplace les fichiers considérés invalides dans le dossier spécifié

-sendOnError
Retransfère les fichiers considérés comme invalides avec R66. Les arguments de lancement du transfert doivent arriver à la fin de la commande, précédés du marqueur `--`.

-ignoreNetworkError
Ignore les erreurs réseau lors de l'envoi de la requête vers le serveur ICAP. Si une se produit, le fichier est considéré comme valide.

-logger `LEVEL`
Spécifie le niveau de logs. Les valeurs possibles sont `DEBUG`, `INFO`, `WARN` et `ERROR`.

Codes de retour :

Code	Signification
0	Le fichier est valide
1	Mauvais arguments dans la ligne de commande
2	Erreur de protocole ICAP
3	Erreur réseau
4	Le fichier est invalide
5	Le fichier est invalide, et une erreur s'est produite durant les post-traitements

Les codes d'erreurs de manières générales indiquent :
— 0 : pour OK
— 1 : pour Warning
— 2 et au-delà : pour une erreur différenciée et bloquante

5.3 Commandes Java brutes

Les commandes Waarp R66 peuvent être appélé directement sans passer par les scripts *Waarp R66 Client* et *Waarp R66 Server*.

Tous les appels sont de la forme :

```
java -cp "path/to/waarp/jars" [options jvm] [classname] [class arguments]
```

Le détail des classes pouvant être exécutées et de leurs arguments est décrit ci-dessous.

5.3.1 Options additionnelles pour Waarp via la JVM

-Dopenr66.locale=en|fr (défaut = en)
Permet choisir entre l'anglais ou le français pour le langage de l'interface

-Dopenr66.ishostproxyfied=1|0 (défaut = 0)
Indique que ce serveur est derrière un proxy (comme R66Proxy ou un matériel équivalent) afin d'empêcher le contrôle des adresses IP de s'appliquer (puisque celle-ci sera celle du Proxy)

-Dopenr66.startup.warning=1|0 (défaut = 1)
Indique si les messages de démarrage doivent être inscrits dans les logs avec le niveau warning (1) ou info (0).

-Dopenr66.startup.checkdb=1|0 Obsolète depuis la version 3.1.0 : Utiliser `-Dopenr66.autoUpgrade` à la place

-Dopenr66.startup.autoUpgrade=0|1 (défaut = 0)
Nouveau dans la version 3.1.0.
Active la mise-à-jour automatique de la base de données au lancement du programme

-Dopenr66.chroot.checked=1|0 (défaut = 1)
Active le mode chroot pour les connexions clientes.
Par exemple, tenter de récupérer un fichier (RECV) depuis un partenaire distant en spécifiant un chemin complet peut être autorisé, même si il est en dehors du répertoire « OUT », sauf si checked=1. Si checked=1, alors tous les fichiers reçus doivent spécifier un chemin inclu dans « OUT », sans remonter au-delà.

-Dopenr66.blacklist.badauthent=1|0 (défaut = 1)
Active le banissement temporaire des partenaires distants en cas d'erreur d'authentification. Cette option permet de prévenir les attaques de type DDOS.
Si `-Dopenr66.ishostproxyfied=1`, alors il est obligatoirement faux. En effet, dans ce cas, si un des partenaires a un problème d'authentification, alors tous les partenaires proxifiés via le même proxy seront bannis puisque visibles depuis la même adresse IP.

-Dopenr66.filename.maxlength=n (défaut = 255)
Définit la longueur maximale autorisée pour les nom de fichiers reçus (nom temporaire et nom final). Ceci n'empêche pas de changer le nom du fichier après et #ORIGINALFILENAME# contient toujours le nom complet d'origine du fichier, non tronqué.

-Dopenr66.trace.stats=n (défaut = 0)

pour mettre en debug certaines traces spécifiques sur des données toutes les n secondes. 0 signifie absence de trace.

-Dopenr66.cache.limit=n et -Dopenr66.cache.timelimit=m (défaut n = 20000, m=180000)

Modifié dans la version 3.2.0 : Le cache a été supprimé, ces options sont ignorées

pour mettre en cache les informations de transfert avec

— n est le nombre maximum de tâches à conserver dans un cache LRU (Last Recent Used). La valeur minimale est 100

— m est le temps maximum en millisecondes avant qu'un élément créé, utilisé ou modifié soit évincé du cache. La valeur minimale est 1000 ms (1s). Une valeur trop grande peut provoquer des consommations mémoires trop importante.

-Dopenr66.usespaceseparator=0|1 (défaut = 0)

Autorise Waarp à utiliser l'espace comme séparateur mais induit des risques de bugs.

-Dopenr66.executebeforetransferred=0|1 (défaut = 1)

Autorise Waarp à exécuter les Error-Tasks si une erreur intervient pendant les « pré-task », avant le transfert effectif

5.3.2 org.waarp.client.Message

Permet d'échanger un message simple avec un partenaire pour s'assurer de la connectivité et de l'authentification respective entre les partenaires.

Cette commande accepte les arguments suivants :

clientConfigurationFile.xml

obligatoire

Fichier de confguration client Waarp R66, en mode synchrone

-to PARTNER

obligatoire

Serveur R66 de destination

-msg MESSAGE

obligatoire pour indiquer le message à transmettre

Contenu du message à transmettre. Celui-ci apparaîtra dans les logs respectifs des deux serveurs (émetteur et récepteur).

Codes de retour :

Code	Signification
0	Succès
1	Les arguments sont incorrects ou le fichier de configuration contient une erreur
2	Une erreur s'est produite lors de la tentative de connexion ou d'authentification

5.3.3 org.waarp.client.BusinessRequest

Permet de déclencher une action à distance avec un partenaire si le partenaire demandeur est autorisé (cf. BUSINESS ROLE)

Cette commande accepte les arguments suivants :

clientConfigurationFile.xml
obligatoire
Fichier de confguration client Waarp R66, en mode synchrone

-to PARTNER
obligatoire
Serveur R66 de destination

-class FULL.CLASS.NAME
obligatoire pour indiquer la classe cible à exécuter de type ExecBusinessTask
Nom de la classe à exécuter

-arg ARGUMENT
Argument à appliquer à la classe

-nolog
Désactive les logs pour ce transfert

Codes de retour :

Code	Signification
0	Succès
2	Les arguments sont incorrects ou le fichier de configuration contient une erreur
N	Une erreur s'est produite lors de la tentative d'exécution

5.4 Exemples

5.4.1 Serveur

Pour démarer le serveur server1

```
waarpr66-server server1 start
```

Arrêter le serveur

Pour arrêter le serveur server1 :

```
waarpr66-server server1 stop
```

5.4.2 Client

Envoie d'un fichier test.file du client client1 au serveur server2 en utilisant la règle default

```
waarpr66-client client1 send -file test.file -to server2 -rule default
```

Envoie des fichier test.file et file.test du client2 au serveur server1 en utilisant la règle prezip en utilisant des blocks de 64b

```
waarpr66-client client2 msend -file test.file,file.test -o server1 -rule prezip -block 64
```

Reprise du transfert id 4 de client3 a serveur8

```
waarpR66-client client3 send -to server8 -id 4
```

CHAPITRE 6

Référence des fichiers de configuration

6.1 server.xml

Le fichier `server.xml` contient les directives de configurations de l'instance serveur.

Note : Les changements dans ce fichier sont pris en compte au redémarrage du serveur.

Les directives de configuration sont réparties en 11 sections :

- *identity* : données concernant l'identité de l'instance
- *server* : données spécifiques au service
- *network* : données concernant les paramètres réseaux
- *ssl* : paramétrage des certificats SSL
- *directory* : dossiers utilisés par le service
- *limit* : paramétrage de l'utilisation des ressources et du comportement interne du serveur
- *db* : paramétrage de la base de données
- *rest* : paramétrage de l'interface REST
- *business* : paramétrage des composantes métiers (Mode `Embedded`)
- *roles* : paramétrage des rôles autorisés des partenaires
- *aliases* : paramétrage d'alias (nom de remplacement) pour des partenaires
- *extendTaskFactory* : paramétrage d'extension de tâches
- *monitor* : paramétrage d'extension du monitoring en mode PUSH REST http(s)

Il existe également des options étendues à la JVM : *ExtraOptions*

6.1.1 Section `identity`

Balise	Type	Obl.	Défaut	Signification
hostid	string	O		Nom de l'hôte en mode non-SSL
sslhostid	string	N		Nom de l'hôte en mode SSL
cryptokey	string	O		Fichier contenant la clef de cryptage des mots de passe stockés, elle-même encryptée en DES
authentfile	string	N		Fichier XML contenant l'authentification des partenaires Waarp R66

6.1.2 Section `server`

Balise	Type	Obl.	Défaut	Signification
serveradmin	string	O		Nom d'utilisateur de l'administrateur utilisé pour accéder à l'interface web d'administration
serverpasswd	string	O		Mot de passe de l'administrateur encryptée avec la clef « cryptokey » utilisé pour accéder à l'interface web d'administration
usenossl	boolean	N	True	Active le mode non-SSL
usessl	boolean	N	False	Active le mode SSL
usehttpcomp	boolean	N	False	Si le mode SSL est activé, active la compression SSL
uselocalexec	boolean	N	False	Par défaut, Waarp R66 utilise System.exec() pour exécuter les processus externes. Cela peut poser des problèmes de performance (limitations de la JDK). L'utilisation de GoldenGate LocalExec Daemon peut permettre d'obtenir de meilleures performance par délégation d'exécution.
lexecadd	string	N	127.0.0.1	Adresse sur laquelle écoute le daemon LocalExec
lexecport	integer	N	9999	Port sur lequel écoute le daemon LocalExec
httpadmin	string	O		Chemin vers le dossier où sont stockées les sources de l'interface d'administration web
admkeypath	string	O		Chemin vers le fichier JKS contenant le certificat HTTPS pour l'interface web d'administration
admkeystorepass	string	O		Mot de passe du fichier JKS contenant le certificat HTTPS pour l'interface web d'administration
admkeypass	string	O		Mot de passe certificat HTTPS pour l'interface web d'administration contenu dans le fichier JKS.
checkaddress	boolean	N	False	Si « True », le serveur R66 vérifie l'adresse IP de l'hôte distant qui demande une connexion
checkclientaddress	boolean	N	False	Si « True », le serveur R66 vérifie l'adresse IP des clients qui demandent une connexion
multiplemonitors	integer	O	1	Nombre de serveurs qui agissent dans le même groupe comme une seule instance R66
pastlimit	integer	N	86400000	Profondeur maximale affichées dans l'interface HTTP de monitoring en ms
minimaldelay	integer	N	5000	Intervalle de rafraîchissement automatique de l'interface HTTP de monitoring en ms
snmpconfig	string	N		Chemin vers le fichier de configuration de l'agent SNMP (voir *référence*)
multiplemoni-	integer	N	1	Nombre d'instances dans un cluster de serveurs Waarp R66

6.1.3 Section `network`

Nouveau dans la version 3.5.0 : Ajout des options `serveraddresses`, `serverssladdresses`, `serverhttpaddresses`, `serverhttpsaddresses`.

Balise	Type	Obl.	Défaut	Signification
serverport	integer	N	6666	Port utilisé pour le protocole R66
serveraddresses	string	N	null	Adresses utilisées pour le protocole R66 (séparées par des virgules)
serversslport	integer	N	6667	Port utilisé pour le protocole R66 en SSL
serverssladdresses	string	N	null	Adresses utilisées pour le protocole R66 en SSL (séparées par des virgules)
serverhttpport	integer	N	8066	Port utilisé pour l'interface web de supervision (la valeur `0` désactive l'interface)
serverhttpaddresses	string	N	null	Adresses utilisées pour l'interface web de supervision (séparées par des virgules)
serverhttpsport	integer	N	8067	Port utilisé pour l'interface web HTTPS d'administration (la valeur `0` désactive l'interface)
serverhttpsaddresses	string	N	null	Adresses utilisées pour l'interface web HTTPS d'administration (séparées par des virgules)

Il est possible de définir avec précision les interfaces (IP) utilisées pour chacun des ports via les options `serveraddresses`, `serverssladdresses`, `serverhttpaddresses`, `serverhttpsaddresses`. Chacune spécifie optionnellement la liste des IP à associer (avec le port défini optionnellement) avec la virgule comme séparateur.

Si cette option n'est pas spécifiée ou vide pour un port de service, toutes les interfaces disponibles seront associées à ce service avec ce port.

Exemple :

```
<network>
  <serverport>6666</serverport>
  <!-- 1 adresse définie en loop -->
  <serveraddresses>127.0.0.1</serveraddresses>
  <serversslport>6667</serversslport>
  <!-- 2 adresses définies -->
  <serverssladdresses>192.168.0.2,10.1.0.10</serverssladdresses>
  <serverhttpport>8066</serverhttpport>
  <!-- Toutes les interfaces seront utilisées, idem si non spécifié -->
  <serverhttpaddresses/>
  <serverhttpsport>8067</serverhttpsport>
  <!-- 1 adresse définie en local -->
  <serverhttpsaddresses>192.168.0.2</serverhttpsaddresses>
</network>
```

```
<network>
  <!-- Toutes les interfaces seront utilisées -->
  <serverport>6666</serverport>
  <serversslport>6667</serversslport>
  <serverhttpport>8066</serverhttpport>
  <serverhttpsport>8067</serverhttpsport>
</network>
```

6.1.4 Section `ssl`

Cette section est optionnelle et peut être omise si le mode SSL est désactivé (`server/usessl` est `false`)

Balise	Type	Obl.	Défaut	Signification
keypath	String	O		Chemin vers le fichier JKS qui contient la clef privée du serveur
keystorepass	String	O		Mot de passe du fichier JKS qui contient la clef privée du serveur
keypass	String	O		Mot de passe de la clef privée du serveur
trustkeypath	String	O		Chemin vers le fichier JKS qui contient la clef publics des hôtes autorisés à se connecter à ce serveur
trustkeystorepass	String	O		Mot de passe du fichier JKS qui contient la clef publics des hôtes autorisés à se connecter à ce serveur
trustuseclientauthenticate	boolean	N	False	Force la connexion des clients en SSL

6.1.5 Section `directory`

Note : Les dossiers par défaut indiqués sont relatifs au dossier `serverhome`.

Balise	Type	Obl.	Défaut	Signification
serverhome	String	O		Chemin vers le répertoire de base du serveur Waarp R66
in	String	N	IN	Chemin du dossier par défaut dans lequel sont déposés les fichiers reçus par défaut (chemin relatif à « serverhome »)
out	String	N	OUT	Chemin du dossier par défaut dans lequel sont pris les fichiers envoyés (chemin relatif à « serverhome »)
arch	String	N	ARCH	Chemin du dossier utilisé pour les archives (chemin relatif à « serverhome »)
work	String	N	WORK	Chemin du dossier utilisé par défaut pour stocker les fichiers en cours de réception (chemin relatif à « serverhome »)
conf	String	N	CONF	Chemin vers le dossier contenant la configuration du serveur

6.1.6 Section `limit`

Nouveau dans la version 3.6.0 : Ajout de l'option `compression` : Active ou Désactive la compression à la volée des blocs transmis, puis en fonction du partenaire.

Balise	Type	Obl.	Défaut	Signification
serverthread	Integer	N	8	Nombre de threads utilisés par les serveur Waarp R66 (valeur recommandée : nombre de cœurs du processeur) (si 0, la valeur sera autmatiquement calculée en fonction)
clientthread	Integer	N	80	Nombre de threads utilisés par le client Waarp R66 (valeur recommandée : serverthread*10) (si 0, la valeur sera autmatiquement calculée en fonction)
memorylimit	Integer	N	1000000000	Quantité maximale de mémoire utilisée pour les services Web et REST (en octets)
sessionlimit	Integer	N	1GB	Bande passante maximale utilisée pour une session (en octets)
globallimit	Integer	N	100GB	Bande passante globale maximale utilisée (en octets)
delaylimit	Integer	N	10000	Délais entre deux vérifications de bande passante. Plus cette valeur est faible, plus le contrôle de la bande passante sera précis. Attention toutefois à ne pas donner de valeur trop faible (en ms)
runlimit	Integer	N	1000	Nombre maximal de transferts actifs simultanés (maximum 50000)
delaycommand	Integer	N	5000	Délais entre deux exécutions du Commander (en ms)
delayretry	Integer	N	30000	Délais entre deux tentatives de transfert en cas d'erreur (en ms)
timeoutcon	Integer	N	30000	Délais de timeout d'une connexion (en ms)
blocksize	Integer	N	65536	Taille de bloc utilisée par le serveur Waarp R66. Une valeur entre 8KB et 16MB est recommandée (en octets)
gaprestart	Integer	N	30	Nombre de blocs écartés lors de la reprise d'un transfert.
usenio	boolean	N	False	Activation du support de NIO pour les fichiers. Selon le JDK, cela peut améliorer les performances.
usecpulimit	boolean	N	False	Utilisation de la limitation de l'utilisation du CPU en jouant sur la bande passante globale pour limiter l'usage des processeurs
usejdkcpulimit	boolean	N	False	Utilisation du support natif du JDK pour contrôler l'utilisation du CPU. Si « False », la librairie Java Sysmon est utilisée
cpulimit	Decimal	N	0.0	Pourcentage maximal d'utilisation du CPU au-delà duquel une demande de transfert est refusée. Les valeurs 0 et 1 désactivent la limite.
connlimit	Integer	N	0	Nombre maximal de connexions. La valeur 0 désactive la limite.
lowcpulimit	decimal	N	0.0	Seuil minimal de consommation de CPU (en pourcentage)
highcpulimit	decimal	N	0.0	Seuil maximal de consommation de CPU (en pourcentage). La valeur 0 désactive le contrôle.
percentde-	de-	N	0.01	Valeur de diminution de la bande passante quand le seuil maximal de consomma-

6.1.7 Section `db`

Note : Si `taskrunnernodb` est à `True`, les autres balises *peuvent* être omises.

Si `taskrunnernodb` est à `False`, où si la balise est absente, toutes les autres balises **doivent** être renseignées.

Balise	Type	Obl.	Défaut	Signification
taskrunnernodb	boolean	N	False	Indique si le serveur utilise une base de données ou non
dbdriver	String	N		Type de base de données utilisé. Sont supportés : oracle, mysql, postgresql, h2
dbserver	String	N		Chaîne de connexion JDBC à la base de données. Consulter le manuel du pilote JDBC utilisé pour la syntaxe exacte.
dbuser	String	N		Utilisateur de la base de données
dbpasswd	String	N		Mot de passe de l'utilisateur de la base de données.
autoUpgrade	boolean	N	True	Vérifie que le modèle de données est à jour au démarrage, et effectue la mise à jour le cas échéant
dbcheck	boolean	N	True	*(déprécié)* Utiliser `autoUpgrade` à la place

6.1.8 Section `rest`

Balise	Type	Obl.	Défaut	Signification
restaddress	string	N		Adresse IP sur laquelle le serveur écoute pour servir l'API REST
serverrestport	integer	N	8068	Port sur lequel le serveur écoute pour servir l'API REST
restssl	boolean	N	False	Active le mode HTTPS pour l'interface REST
restauthenticated	boolean	N	False	Active l'authentification des requêtes vers l'API REST
resttimelimit	integer	N	-1	Active la limitation de validité dans le temps des requêtes (en ms). `-1` désactive cette limitation.
restsignature	boolean	N	True	Active la signature des requêtes REST
restsigkey	string	N		Chemin vers le fichier contenant la clef de signature des requêtes REST (cf. *restsignkey*)
restmethod		O		Voir ci-dessous.

Les balises `restmethod` peuvent être renseignées plusieurs fois. Elles permettent d'activer chaque fonctionnalités de l'API REST individuellement.

Chaque ocurrence de `restmethod` doit contenir deux balises :

— `restname` : le nom de la fonctionnalité à paramétrer (plusieurs fonctionnalités peuvent être renseignées, séparées par des espaces)

— `restcrud` : les actions actives pour la (les) fonctionnalités en question.

Par exemple :

```
<restmethod>
     <restname>ALL</restname>
     <restcrud>R</restcrud>
</restmethod>
<restmethod>
   <restname>DbHostAuth DbRule</restname>
   <restcrud>CRU</restcrud>
</restmethod>
<restmethod>
   <restname>Bandwidth</restname>
   <restcrud>RU</restcrud>
</restmethod>
```

Les fonctionnalités sont les suivantes :

Fonctionnalité	Description
All	Alias regroupant toutes les fonctionnalités ci-dessous
DbTaskRunner	Actions sur les transferts
DbHostAuth	Actions sur la liste des partenaires
DbRule	Actions sur les règles de transfert
DbHostConfiguration	Actions sur la configuration des hôtes
DbConfiguration	Actions sur les limitations de bandes passantes
Bandwidth	Actions sur les limitations de bandes passantes
Business	Actions sur l'intégration métier
Config	Import/export de la configuration
Information	Récupère des informations sur les transferts
Log	Actions sur les logs
Server	Actions sur le serveur
Control	Actions sur les transferts

Pour chaque fonctionnalités, les actions à activer sont indiquées par une combinaison des lettres `C`, `R`, `U` et `D` (`C` pour *création*, `R` pour *lecture*, `U` pour *mise-à-jour* et `D` pour *suppression*) ou seules les actions voulues doivent être indiquées.

6.1.9 Section `business`

Balise	Type	Obl.	Défaut	Signification
businessid	string	N		Id d'un partenaire autorisé à déclencher des opérations Business

6.1.10 Section `roles`

Il s'agit d'une liste de `role`, contenant chacun :

Balise	Type	Obl.	Défaut	Signification
roleid	string	O		Id d'un partenaire
roleset	string	O		liste de rôles autorisés, séparés par un « blanc » ou un « | », parmi : NOACCESS,READONLY,TRANSFER,RULE,HOST,LIMIT,SYSTEM,LOGCONTROL,PARTNER(READONL

6.1.11 Section `aliases`

Il s'agit d'une liste de `alias`, contenant chacun :

Balise	Type	Obl.	Défaut	Signification
realid	string	O		Id d'un partenaire
aliasid	string	O		liste de noms alias équiavelents, séparés par un « blanc » ou un « | »

6.1.12 Section `ExtendTaskFactory`

Nouveau dans la version 3.6.0 : Ajout du sous-ensemble `extendTaskFactory` qui contient l'option `extendedtaskfactories` : pour la Factory `org.waarp.openr66.s3.taskfactory.S3TaskFactory`, si la classe est dans le claspath, il n'est pas nécessaire de l'ajouter.

Balise	Type	Obl.	Défaut	Signification
extendedtaskfactories	String	N	vide	Liste (séparée par des virgules) des TaskFactory en tant qu'extension pour ajouter des tâches à WaarpR66

6.1.13 Section `pushMonitor`

Cette section décrit comment monitorer R66 via des appels REST HTTP(s) vers un serveur tiers (en mode PUSH).

Nouveau dans la version 3.6.0 : Ajout du sous-ensemble `pushMonitor` qui contient les options communes `url`, `delay`, `intervalincluded`, `transformlongasstring`, `token`, `apiKey`, les options spécifiques``endpoint``, `keepconnection` et `basicAuthent` sont liées à une API REST en destination, les options spécifiques``index``, `prefix`, `username`, `paswd` et `compression` sont liées à Elasticsearch en destination.

Balise	Type	Obl.	Défaut	Signification
Partie commune				
url	string	N	null	URL de base pour les exports du moniteur en mode POST HTTP(S) JSON
delay	integer	N	1000	Délai entre deux vérifications de changement de statuts sur les transferts
intervalincluded	boolean	N	True	Si « True », les informations de l'intervalle utilisé seront fournies
transformlongasstring	boolean	N	False	Si « True », les nombres « long » seront convertis en chaîne de caractères, sinon ils seront numériques
token	string	N	null	Spécifie si nécessaire le token dans le cadre d'une authentification via Token
apiKey	string	N	null	Spécifie si nécessaire le password dans le cadre d'une authentification via ApiKey (format `apiId:apiKey`)
Partie API REST				
endpoint	string	N	null	End point à ajouter à l'URL de base
keepconnection	boolean	N	True	Si « True », la connexion HTTP(S) sera en Keep-Alive (pas de réouverture sauf si le serveur la ferme), sinon la connexion sera réinitialisée pour chaque appel
basicAuthent	string	N	null	Spécifie si nécessaire l'authentification basique
Partie Elasticsearch				
index	string	N	null	Contient le nom de l'index avec de possibles substitutions, dont `%%WARPHOST%%` pour le nom du host concerné, et les `%%DATETIME%%`, `%%DATEHOUR%%`, `%%DATE%%`, `%%YEARMONTH%%`, `%%YEAR%%` pour des substitutions de date et heure partiellement (`yyyy.MM.dd.HH.mm` à `yyyy`)
prefix	string	N	null	Spécifie si nécessaire un prefix global dans le cas d'usage d'un Proxy devant Elasticsearch
username	string	N	null	Spécifie si nécessaire le username (et son password) dans le cadre d'une authentification basique
paswd	string	N	null	Spécifie si nécessaire le password dans le cadre d'une authentification basique
compression	boolean	N	True	Spécifie si les flux sont compressés (par défaut True)

Voir aussi :

Une documentation complète de la configuration du monitoring en mode export REST HTTP(S) ou en mode Elasticsearch (JRE >= 8) est disponible *ici*

6.1.14 Section ExtraOptions

Mise à jour automatique de la base de données

Par défaut, le champ <root><version>version</version></root> du fichier de configuration XML est géré par Waarp pour vérifier la configuration de la base de données et sa version par rapport à celle du programme, afin de permettre une mise à jour automatique.

Cette mise à jour automatique peut être empêchée par l'option <db><autoUpgrade>False</autoUpgrade>...</db> ou grâce à la propriété Java -Dopenr66.startup.dbcheck=0.

Partage d'une même base entre plusieurs moniteurs Waarp

Dans le cas où une base est partagée entre plusieurs moniteurs R66, afin d'être capable de voir tous les transferts dans la console web d'administration, vous pouvez indiquer une option spéciale dans « Autres informations » avec l'identifiant qui sera utilisé pour se connecter à cette interface Web.

```
<root>...<seeallid>id1,id2,...,idn</seeallid></root>
```

6.1.15 Exemple complet

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:x0="http://www.w3.org/2001/XMLSchema">
    <comment>Configuration file for a server with a Postgresql database</comment>
    <identity>
        <hostid>monserveur</hostid>
        <sslhostid>monserveur-ssl</sslhostid>
        <cryptokey>/etc/waarp/cryptokey.des</cryptokey>
    </identity>
    <server>
        <serveradmin>admin</serveradmin>
        <serverpasswd>5a4b7c6a66065cbb622acefec8c3a302</serverpasswd>
        <usenossl>True</usenossl> <!-- Might be False if not needed -->
        <usessl>True</usessl> <!-- Might be False if not needed -->
        <usehttpcomp>False</usehttpcomp>
        <uselocalexec>False</uselocalexec>
        <httpadmin>/etc/waarp/admin</httpadmin>
        <admkeypath>/etc/waarp/adminkey.jks</admkeypath>
        <admkeystorepass>password</admkeystorepass>
        <admkeypass>password</admkeypass>
        <checkaddress>False</checkaddress>
        <checkclientaddress>False</checkclientaddress>
        <pastlimit>86400000</pastlimit>
        <minimaldelay>5000</minimaldelay>
        <multiplemonitors>1</multiplemonitors>
        <!-- Might be removed if not needed -->
        <snmpconfig>/etc/waarp/snmpconfig.xml</snmpconfig>
    </server>
    <network>
        <serverport>6666</serverport>
        <serversslport>6667</serversslport>
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <serverhttpport>8066</serverhttpport>
        <serverhttpsport>8067</serverhttpsport>
    </network>
    <ssl>
        <keypath>/etc/waarp/key.jks</keypath>
        <keystorepass>password</keystorepass>
        <keypass>password</keypass>
        <trustkeypath>/etc/waarp/trustkey.jks</trustkeypath>
        <trustkeystorepass>password</trustkeystorepass>
        <trustuseclientauthenticate>True</trustuseclientauthenticate>
    </ssl>
    <directory>
        <serverhome>/var/lib/waarp</serverhome>
        <in>in</in>
        <out>out</out>
        <arch>arch</arch>
        <work>work</work>
        <conf>conf</conf>
    </directory>
    <rest>
        <restaddress>0.0.0.0</restaddress>
        <restport>8088</restport>
        <restssl>true</restssl>
        <restauthenticated>true</restauthenticated>
        <resttimelimit>3000</resttimelimit>
        <restsignature>true</restsignature>
        <restsigkey>/etc/waarp/restsigning.key</restsigkey>
        <restmethod>
            <restname>ALL</restname>
            <restcrud>CRUD</restcrud>
        </restmethod>
        <restmethod>
           <restname>Bandwidth</restname>
           <restcrud>CRUD</restcrud>
        </restmethod>
        <restmethod>
           <restname>Information</restname>
           <restcrud>CRUD</restcrud>
        </restmethod>
        <restmethod>
           <restname>Server</restname>
           <restcrud>CRUD</restcrud>
        </restmethod>
        <restmethod>
           <restname>Control</restname>
           <restcrud>CRUD</restcrud>
        </restmethod>
    </rest>
    <limit>
        <!-- Might be changed to number of cores -->
        <serverthread>8</serverthread>
        <!-- Might be changed to number of cores x 10 -->
```

(suite sur la page suivante)

(suite de la page précédente)

```
        <clientthread>80</clientthread>
        <usefastmd5>False</usefastmd5>
        <timeoutcon>10000</timeoutcon>
        <delayretry>10000</delayretry>
        <!-- Might be changed to 100000 -->
        <memorylimit>1000000</memorylimit>
        <!-- Might be changed to 100 to 1000 according to activity -->
        <runlimit>1000</runlimit>
    </limit>
    <db>
        <dbdriver>postgresql</dbdriver>
        <dbserver>jdbc:postgresql://localhost:5432/waarp_r66</dbserver>
        <dbuser>username</dbuser>
        <dbpasswd>password</dbpasswd>
        <autoUpgrade>false</autoUpgrade>
    </db>
    <extendTaskFactory>
        <extendedtaskfactories>org.waarp.openr66.s3.taskfactory.S3TaskFactory</
↪extendedtaskfactories>
    </extendTaskFactory>
    <pushMonitor>
        <url>http://127.0.0.1:8999</url>
        <endpoint>/log</endpoint>
        <delay>1000</delay>
        <keepconnection>true</keepconnection>
        <intervalincluded>true</intervalincluded>
        <transformlongasstring>false</transformlongasstring>
    </monitor>
</config>
```

6.2 client.xml

Le fichier `client.xml` contient les directives de configurations de l'instance cliente.

Les directives de configuration sont réparties en 7 sections :
- *identity* : données concernant l'identité de l'instance
- *ssl* : paramétrage des certificats SSL
- *directory* : dossiers utilisés par le service
- *limit* : paramétrage de l'utilisation des ressources et du comportement interne du serveur
- *db* : paramétrage de la base de données
- *extendTaskFactory* : paramétrage d'extension de tâches

6.2.1 Section `identity`

Balise	Type	Obl.	Défaut	Signification
hostid	string	O		Nom de l'hôte en mode non-SSL
sslhostid	string	N		Nom de l'hôte en mode SSL
cryptokey	string	O		Fichier contenant la clef de cryptage des mots de passe stockés, elle-même encryptée en DES
authentfile	string	N		Fichier XML contenant l'authentification des partenaires Waarp R66

6.2.2 Section `client`

Balise	Type	Obl.	Défaut	Signification
taskrunnernodb	boolean	N	False	Indique que le client n'utilise pas de base de données
businessfactorynetwork	string	N	null	Indique la classe Factory pour les comportements « métiers » à associer à Waarp (Embedded)

6.2.3 Section `ssl`

Cette section est optionelle et peut être omise si le mode SSL n'est pas utilisé.

Balise	Type	Obl.	Défaut	Signification
keypath	String	O		Chemin vers le fichier JKS qui contient la clef privée du serveur
keystorepass	String	O		Mot de passe du fichier JKS qui contient la clef privée du serveur
keypass	String	O		Mot de passe de la clef privée du serveur
trustkeypath	String	O		Chemin vers le fichier JKS qui contient la clef publics des hôtes autorisés à se connecter à ce serveur
trustkeystorepass	String	O		Mot de passe du fichier JKS qui contient la clef publics des hôtes autorisés à se connecter à ce serveur
trustuseclientauthenticate	boolean	N	False	Force la connexion des clients en SSL

6.2.4 Section `directory`

Note : Les dossiers par défaut indiqués sont relatifs au dossier `serverhome`.

Balise	Type	Obl.	Défaut	Signification
serverhome	String	O		Chemin vers le répertoire de base du serveur Waarp R66
in	String	N	IN	Chemin du dossier par défaut dans lequel sont déposés les fichiers reçus par défaut (chemin relatif à « serverhome »)
out	String	N	OUT	Chemin du dossier par défaut dans lequel sont pris les fichiers envoyés (chemin relatif à « serverhome »)
arch	String	N	ARCH	Chemin du dossier utilisé pour les archives (chemin relatif à « serverhome »)
work	String	N	WORK	Chemin du dossier utilisé par défaut pour stocker les fichiers en cours de réception (chemin relatif à « serverhome »)
conf	String	N	CONF	Chemin vers le dossier contenant la configuration du serveur

6.2.5 Section `limit`

Balise	Type	Obl.	Défaut	Signification
serverthread	Integer	N	8	Nombre de threads utilisés par les serveur Waarp R66 (valeur recommandée : nombre de cœurs du processeur)
clientthread	Integer	N	80	Nombre de threads utilisés par le client Waarp R66 (valeur recommandée : serverthread*10)
memorylimit	Integer	N	1000000000	Quantité maximale de mémoire utilisée pour les services Web et REST (en octets)
sessionlimit	Integer	N	1GB	Bande passante maximale utilisée pour une session (en octets)
globallimit	Integer	N	100GB	Bande passante globale maximale utilisée (en octets)
delaylimit	Integer	N	10000	Délais entre deux vérifications de bande passante. Plus cette valeur est faible, plus le contrôle de la bande passante sera précis. Attention toutefois à ne pas donner de valeur trop faible (en ms)
runlimit	Integer	N	1000	Nombre maximal de transferts actifs simultanés (maximum is 50000)
delaycommand	Integer	N	5000	Délais entre deux exécutions du Commander (en ms)
delayretry	Integer	N	30000	Délais entre deux tentatives de transfert en cas d'erreur (en ms)
timeoutcon	Integer	N	30000	Délais de timeout d'une connexion (en ms)
blocksize	Integer	N	65536	Taille de bloc utilisée par le serveur Waarp R66. Une valeur entre 8KB et 16MB est recommandée (en octets)
gaprestart	Integer	N	30	Nombre de blocs écartés lors de la reprise d'un transfert.
usenio	boolean	N	False	Activation du support de NIO pour les fichiers. Selon le JDK, cela peut améliorer les performances.
usecpulimit	boolean	N	False	Utilisation de la limitation de l'utilisation du CPU au démarrage d'une requête
usejdkcpulimit	boolean	N	False	Utilisation du support natif du JDK pour contrôler l'utilisation du CPU. Si « False », la librairie Java Sysmon est utilisée
cpulimit	Decimal	N	0.0	Pourcentage maximal d'utilisation du CPU au-delà duquel une demande de transfert est refusée. Les valeurs 0 et 1 désactivent la limite.
connlimit	Integer	N	0	Nombre maximal de connexions. La valeur 0 désactive la limite.
lowcpulimit	decimal	N	0.0	Seuil minimal de consommation de CPU (en pourcentage)
highcpulimit	decimal	N	0.0	Seuil maximal de consommation de CPU (en pourcentage). La valeur 0 désactive le contrôle.

6.2.6 Section db

Note : Si taskrunnernodb est à True, les autres balises *peuvent* être omises.

Si taskrunnernodb est à False, où si la balise est absente, toutes les autres balises **doivent** être renseignées.

Balise	Type	Obl.	Défaut	Signification
taskrunnernodb	boolean	N	False	Indique si le serveur utilise une base de données ou non
dbdriver	String	N		Type de base de données utilisé. Sont supportés : oracle, mysql, postgresql, h2
dbserver	String	N		Chaîne de connexion JDBC à la base de données. Consulter le manuel du pilote JDBC utilisé pour la syntaxe exacte.
dbuser	String	N		Utilisateur de la base de données
dbpasswd	String	N		Mot de passe de l'utilisateur de la base de données.
autoUpgrade	boolean	N	True	Vérifie que le modèle de données est à jour au démarrage, et effectue la mise à jour le cas échéant
dbcheck	boolean	N	True	*(déprécié)* Utiliser autoUpgrade à la place

6.2.7 Section ExtendTaskFactory

Nouveau dans la version 3.6.0 : Ajout du sous-ensemble extendTaskFactory qui contient l'option extendedtaskfactories : pour la Factory org.waarp.openr66.s3.taskfactory.S3TaskFactory, si la classe est dans le claspath, il n'est pas nécessaire de l'ajouter.

Balise	Type	Obl.	Défaut	Signification
extendedtaskfactories	String	N	vide	Liste (séparée par des virgules) des TaskFactory en tant qu'extension pour ajouter des tâches à WaarpR66

6.2.8 Exemple complet

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:x0="http://www.w3.org/2001/XMLSchema">
  <comment>Client configuration template</comment>
  <identity>
     <hostid>monserveur</hostid>
     <sslhostid>monserveur-ssl</sslhostid>
     <cryptokey>/etc/waarp/cryptokey.des</cryptokey>
  </identity>
  <client/>
  <ssl>
     <keypath>/etc/waarp/key.jks</keypath>
     <keystorepass>password</keystorepass>
     <keypass>password</keypass>
     <trustkeypath>/etc/waarp/trustkey.jks</trustkeypath>
     <trustkeystorepass>password</trustkeystorepass>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <trustuseclientauthenticate>True</trustuseclientauthenticate>
  </ssl>
  <directory>
        <serverhome>/var/lib/waarp</serverhome>
        <in>in</in>
        <out>out</out>
        <arch>arch</arch>
        <work>work</work>
        <conf>conf</conf>
  </directory>
  <limit>
      <serverthread>8</serverthread>
      <clientthread>80</clientthread>
      <usefastmd5>False</usefastmd5>
      <timeoutcon>10000</timeoutcon>
      <delayretry>10000</delayretry>
  </limit>
  <db>
        <dbdriver>postgresql</dbdriver>
        <dbserver>jdbc:postgresql://localhost:5432/waarp_r66</dbserver>
        <dbuser>username</dbuser>
        <dbpasswd>password</dbpasswd>
        <autoUpgrade>false</autoUpgrade>
  </db>
 <extendTaskFactory>
   <extendedtaskfactories>org.waarp.openr66.s3.taskfactory.S3TaskFactory</
↪extendedtaskfactories>
 </extendTaskFactory>
</config>
```

6.2.9 Exemple complet minimaliste pour empreinte mémoire minimale

Launching the client using the option `-Xmx128m` on command line option, in addition to the following example configuration file for the client only.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:x0="http://www.w3.org/2001/XMLSchema">
  <comment>Client configuration template</comment>
  <identity>
     <hostid>monserveur</hostid>
     <sslhostid>monserveur-ssl</sslhostid>
     <cryptokey>/etc/waarp/cryptokey.des</cryptokey>
     <usenossl>True</usenossl> <!-- Might be False if not needed -->
     <usessl>True</usessl> <!-- Might be False if not needed -->
  </identity>
  <client/>
  <ssl>
     <keypath>/etc/waarp/key.jks</keypath>
     <keystorepass>password</keystorepass>
     <keypass>password</keypass>
     <trustkeypath>/etc/waarp/trustkey.jks</trustkeypath>
```

(suite sur la page suivante)

(suite de la page précédente)

```
    <trustkeystorepass>password</trustkeystorepass>
    <trustuseclientauthenticate>True</trustuseclientauthenticate>
  </ssl>
  <directory>
        <serverhome>/var/lib/waarp</serverhome>
        <in>in</in>
        <out>out</out>
        <arch>arch</arch>
        <work>work</work>
        <conf>conf</conf>
  </directory>
  <limit>
      <serverthread>1</serverthread>
      <clientthread>1</clientthread>
      <runlimit>1</runlimit>
      <usefastmd5>False</usefastmd5>
      <timeoutcon>10000</timeoutcon>
      <delayretry>10000</delayretry>
  </limit>
  <db>
        <dbdriver>postgresql</dbdriver>
        <dbserver>jdbc:postgresql://localhost:5432/waarp_r66</dbserver>
        <dbuser>username</dbuser>
        <dbpasswd>password</dbpasswd>
        <autoUpgrade>false</autoUpgrade>
  </db>
</config>
```

6.3 snmpconfig.xml

Le fichier `server.xml` contient la configuration des serveurs snmpconfig avec lesquels une instance peut communiquer.

Note : Les changements dans ce fichier sont pris en compte au redémarrage du serveur.

Les directives de configuration sont réparties en 3 sections :
- *config* : paramétrage du système SNMP de l'instance Waarp
- *targets* : liste des serveurs SNMP à utiliser
- *securities* : données d'authentification SNMP

6.3.1 Section `config`

Balise	Type	Obl.	Défaut	Signification
localaddress	string	O		Adresse IP sur laquelle le service SNMP doit écouter au format `udp:address/port` ou `tcp:address/port` (peut être renseigné plusieurs fois)
nbthread	string	N	4	Nombre de threads à utiliser pour le serveur SNMP
filtered	boolean	N	False	Active le filtrage des connexions SNMPv1 or SNMPv2c entrantes sur l'adresse IP du client.
usetrap	string	N	True	Utilise des messages « traps » (`True`) ou « inform » (`False`) Lors de l'envoi d'événement à un serveur SNMP
trapinformlevel	integer	N	0	Défini le niveau de criticité des messages à envoyer : 0 : Aucun ; 1 : Démarrage/Arrêt du serveur ; 2 : messages de niveau critique ; 3 : messages de niveau erreur ; 4 : messages de niveau warning ; 5 : tous les événements

6.3.2 Section `targets`

La section `targets` regroupe la liste des serveurs SNMP auxquels envoyer des événements.

Chaque serveur est défini dans un bloc XML `target` acceptant les balises suivantes (voir *Exemple complet*) :

Balise	Type	Obl.	Défaut	Signification
name	string	O		Nom à utiliser pour le serveur
domain	string	N	UdpIpv4	Domaine à utiliser pour le serveur. Doit être une des valeurs suivantes : `UdpIpV4`, `UdpIpv6`, `UdpIpV4e`, `UdpIpv6z`, `TcpIpv4`, `TcpIpv6`, `TcpIpv4z`, `TcpIpv6z`
address	string	O		Adresse du serveur au format `adress/port`
timeout	integer	N	200	Délais maximum d'attente de réponse pour les messages « inform » (en ms)
retries	integer	N	1	Nombre de tentative d'envoi des messages « inform » non acquittés par le serveur
isv2	boolean	N	True	Défini si les serveur utilise le protocole SNMPv2c (`True`) ou SNMPv3 (`False`)

6.3.3 Section `securities`

La section `securities` défini les paramètres de sécurité pour SNMPv3. Plusieurs profils peuvent être définis.

Chaque profil est défini dans un bloc XML `security` acceptant les balises suivantes (voir *Exemple complet*) :

Balise	Type	Obl.	Défaut	Signification
security-name	string	O		Nom à utiliser
securityauth-protocol	string	N		Protocole à utiliser pour l'authentification. Doit être une des valeurs suivantes : `MD5`, `SHA`
securityauth-pass	string	N		Mot de passe pour l'authentification (peut être vide)
securitypriv-protocol	string	N		Protocole à utiliser pour le chiffrement. Doit être une des valeurs suivantes : `P3DES`, `PAES128`, `PAES192`, `PAES256`, `PDES`
securitypriv-pass	string	N		Mot de passe pour le chiffrement (peut être vide)

6.3.4 Exemple complet

```
<?xml version="1.0" encoding="UTF-8"?>
<snmpconfig xmlns:x0="http://www.w3.org/2001/XMLSchema">
   <config>
      <localaddress>udp:0.0.0.0/2001</localaddress>
      <localaddress>tcp:0.0.0.0/2002</localaddress>
      <nbthread>4</nbthread>
      <filtered>False</filtered>
      <usetrap>True</usetrap>
      <trapinformlevel>4</trapinformlevel>
   </config>
   <targets>
      <target>
         <name>notificationV2c</name>
         <domain>UdpIpv4</domain>
         <address>127.0.0.1/162</address>
         <timeout>200</timeout>
         <retries>1</retries>
         <isv2>True</isv2>
      </target>
      <target>
         <name>notificationV3</name>
         <domain>UdpIpv4</domain>
         <address>127.0.0.1/162</address>
         <timeout>200</timeout>
         <retries>1</retries>
         <isv2>False</isv2>
      </target>
   </targets>
   <securities>
      <security>
         <securityname>SHADES</securityname>
         <securityauthprotocol>SHA</securityauthprotocol>
         <securityauthpass>SHADESAuthPassword</securityauthpass>
         <securityprivprotocol>PDES</securityprivprotocol>
         <securityprivpass>SHADESPrivPassword</securityprivpass>
      </security>
```

(suite sur la page suivante)

(suite de la page précédente)

```
   </securities>
</snmpconfig>
```

6.4 rule.xml

comment
String

idrule
nonEmptyString
ID de la règle

hostids
List of Host Ids allowed to use this rule. No Host Id means all allowed
[hostid]

hostid
nonEmptyString
Hostname des partenaires authorisés à utiliser cette règle

```
<rule>
  ...
  <hostids>
    <hostid>server1</hostid>
    <hostid>server2</hostid>
    ...
  </hostids>
  ...
<rule>
```

mode
nonNulInteger
1=SEND 2=RECV 3=SEND+MD5 4=RECV+MD5 5=SENDTHROUGHMODE 6=RECVTHROUGHMODE 7=SENDMD5THROUGHMODE 8=RECVMD5THROUGHMODE

recvpath
token (IN)
Dossier de reception

sendpath
token (OUT)
Dossier d'envoi

archivepath
token (ARCH)
Dossier d'archive

workpath
token (WORK)
Dossier de travail

rpretasks

Tasks
[Task]
List des tâche à exécuter avant le transfert par l'envoyeur

rposttasks
Tasks
[Task]
List des tâche à exécuter après le transfert par l'envoyeur

rerrortasks

Tasks
[Task]
List des tâche à exécuter en cas d'erreur du transfert par l'envoyeur

spretasks

Tasks
[Task]
List des tâche à exécuter avant le transfert par le receveur

sposttasks

Tasks
[Task]
List des tâche à exécuter après le transfert par le receveur

serrortasks

Tasks
[Task]
List des tâche à exécuter en cas d'erreur le transfert par le receveur[3]

Les blocs <*task*> definissent les tâches opérées par les différents acteurs de la règle

type
nonEmptyString
Type de tâche : LOG, SNMP, MOVE, MOVERENAME, COPY, COPYRENAME, LINKRENAME, RENAME, DELETE, VALIDFILEPATH, EXEC, EXECMOVE, EXECOUTPUT, EXECJAVA, RESTART, TRANSFER, RESCHEDULE, FTP, TAR, ZIP, TRANSCODE, UNZEROED, CHKFILE, CHMOD, ICAP

path
nonEmptyString
Argument -généralement un path- appliqué à la tâche, des substitutions sont possibles #TRUEFULLPATH#, #FILESIZE#, #RULE#, #DATE#, #TRANSFERID#, ..."

delay
nonNegInteger
Delai (ms) maximum pour l'execution de la tâche

6.5 Type de Task

6.5.1 Format général d'une tâche

Nouveau dans la version 3.6.0 : Ajout de l'option #COMPRESS# qui, avec la configuration du serveur compression à Vrai, permet de compresser un transfert en accord avec le partenaire (qui peut ne pas supporter la compression et donc celle-ci sera non activée pour ce transfert).

Une tâche est définie selon un format unifié XML :

```
<tasks>
 <task>
  <type>NAME</type>
  <path>path</path>
  <delay>x</delay>
  <rank>n</rank>
 </task>
</tasks>
```

- Type est l'identifiant du type de tâche à exécuter (les types sont présentés ci-après.
- Path est un argument fixé par la règle et dont des remplacements de mots clefs sont opérés :
  - #TRUEFULLPATH# : Chemin complet du fichier courant
  - #TRUEFILENAME# : Nom du fichier courant (hors chemin) (différent côté réception)
  - #ORIGINALFULLPATH# : Chemin complet du fichier d'origine (avant changement côté réception)
  - #ORIGINALFILENAME# : Nom du fichier d'origine (avant changement côté réception)
  - #FILESIZE# : Taille du fichier s'il existe
  - #INPATH# : Chemin du dossier de réception
  - #OUTPATH# : Chemin du dossier d'émission
  - #WORKPATH# : Chemin du dossier de travail
  - #ARCHPATH# : Chemin du dossier d'archive
  - #HOMEPATH# : Chemin du dossier du répertoire « racine » de Waarp
  - #RULE# : Règle utilisé pour le transfert
  - #DATE# : Date courante au format yyyyMMdd
  - #HOUR# : Heure courante au format HHmmss
  - #REMOTEHOST# : Nom DNS du partenaire
  - #REMOTEHOSTIP# : IP du partenaire
  - #LOCALHOST# : Nom DNS local du serveur Waarp
  - #LOCALHOSTIP# : IP du serveur Waarp
  - #TRANSFERID# : Identifiant de transfert
  - #REQUESTERHOST# : Nom du partenaire initiateur du transfert
  - #REQUESTEDHOST# : Nom du partenaire recevant la demande de transfert
  - #FULLTRANSFERID# : Identifiant complet du transfert comme TRANSFERID_REQUESTERHOST_REQUESTEDHOST
  - #RANKTRANSFER# : Rang du bloc courant ou final du fichier transféré
  - #BLOCKSIZE# : Taille du bloc utilisé
  - #ERRORMSG# : Le message d'erreur courant ou « NoError » si aucune erreur n'a été levée jusqu'à cet appel
  - #ERRORCODE# : Le code erreur courant ou - (Unknown) si aucune erreur n'a été levée jusqu'à cet appel
  - #ERRORSTRCODE# : Le message lié au code d'erreur courant ou Unknown si aucune erreur n'a été levée jusqu'à cet appel
  - #NOWAIT# : Utilisé par la tâche EXEC pour spécifier que la commande est à exécuter en mode asynchrone, sans en attendre le résultat
  - #LOCALEXEC# : Utilisé par la tâche EXEC pour spécifier que la commande est à exécuter de manière distante (pas dans la JVM courante) mais au travers d'un démon LocalExec (spécifié dans la configuration globale
  - COMPRESS : Utilisé en complément de l'option compression dans la configuration du serveur pour indiquer la demande de pouvoire compresser par bloc un transfert (si le partenaire ne dispose pas de l'option de

compression, cela sera refusé).

Par exemple, un `Path` défini comme : `some #DATE# some2 #TRANSFERID# some3 #REMOTEHOST# some4` donnera `some 20130529 some2 123456789123 some3 remotehostid some4`

- `Delay` est généralement le délai (si précisé) maximum pour l'exécution d'une tâche avant qu'elle tombe en erreur pour dépassement de délai.
- `Rank` est optionnel et permet d'indiquer un rang d'exécution des tâches, laissant une souplesse sur l'ordre d'écriture en période de tests. Il n'est pas conseillé de l'utiliser par défaut hormis pour les tests et validation de plusieurs tâches.

De plus, une tâche utilisera les arguments de transfert eux-même (`Transfer Information`) pour déduire les paramètres finaux en utilisant la fonction `String.format(path règle, info transfert.éclaté en sous chaînes via le séparateur " ")`

L'argument de transfert (spécifié par `-info` dans les commandes de transferts) ou le `path` peuvent contenir une MAP au format JSON qui intègre des éléments utiles aux tâches ou au fonctionnement de Waarp (comme le `DIGEST`, le `RESCHDEDULE`, l'option `follow`).

Ceci permet de rendre les arguments très adaptatifs.

Par exemple :

- si la règle définie `Path` comme `some %s some2 %d some3 %s some4`
- et si les informations de transferts sont `info1 1 info2`
- Le résultat sera pour cette tâche : `some info1 some2 1 some3 info2 some4`

6.5.2 Tâches informatives

LOG

Cette tâche loggue ou écrit dans un fichier externe des informations :

- si `delay` est 0, aucune sortie ne sera effectuée
- si `delay` est 1, les informations seront envoyées vers un log
- si `delay` est 2, les informations seront envoyées dans un fichier (le dernier argument sera le chemin complet du fichier de sortie)
- si `delay` est 3, les informations seront envoyées dans le log et dans un fichier (le dernier argument sera le chemin complet du fichier de sortie)

Si le premier mot de ce log est un parmi `debug`, `info`, `warn` ou `error`, ce sera le niveau du log utilisé.

Exemple :

```
<task>
  <type>LOG</type>
  <path>warn information /path/logfile</path>
  <delay>2</delay>
</task>
```

Ceci logguera un log « WARN » dans le fichier `/path/logfile` sans trace dans les logs usuels.

SNMP

Cette tâche émet un trap SNMP :
- si `delay` est 0, un trap SNMP warning/info est envoyé avec le champ info et le transfer ID
- si `delay` est 1, un trap SNMP/info avec toutes les informations de transfert sont envoyées

Si le premier mot de ce log est un parmi `debug`, `info`, `warn` ou `error`, ce sera le niveau du log utilisé.

Exemple :

```
<task>
  <type>SNMP</type>
  <path>information</path>
  <delay>0</delay>
</task>
```

Ceci enverra un trap SNMP/info contenant `information` et le TransferID.

6.5.3 Tâches agissant sur l'emplacement du fichier

COPY

Copie le fichier au chemin désigné comme argument sans renommer le fichier (même nom de base). Le chemin obtenu sera un chemin absolu (et non un chemin relatif).
- `Delay` et Transfer Information sont ignorés.
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>COPY</type>
  <path>/newpath/</path>
  <delay/>
</task>
```

Cela copiera le fichier courant vers `/newpath/` en tant que `/newpath/currentfilename`. Le fichier courant reste le même (inchangé).

COPYRENAME

Copie le fichier au chemin désigné comme argument en renommant le fichier. Le chemin obtenu sera un chemin absolu (et non un chemin relatif).
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`). Le chemin obtenu doit être un chemin absolu.
- `Delay` est ignoré.
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>COPYRENAME</type>
  <path>/newpath/newfilename_%s_#TRANSFERID#</path>
  <delay/>
</task>
```

Si le Transfer Information est myinfoFromTransfer, cela copiera le fichier dans un nouveau fichier nommé /newpath/newfilename_myinfoFromTransfer_transferid où transferid sera remplacé par un identifiant unique (comme 123456789). Le fichier courant reste le même (inchangé).

MOVE

Déplace le fichier au chemin désigné comme argument sans renommer le fichier (même nom de base). Le chemin obtenu sera un chemin absolu (et non un chemin relatif).

- Une fois le Path transformé selon les remplacements dynamiques, il est utilisé comme String Format avec le Transfer Information utilisé en entrée (String.format(Path,Info)). Le chemin obtenu doit être un chemin absolu.
- Delay est ignoré.
- Le fichier est marqué comme déplacé.

Exemple :

```
<task>
  <type>MOVE</type>
  <path>/newpath/</path>
  <delay/>
</task>
```

Le fichier sera déplacé (non copié) dans le répertoire /newpath/. Le fichier courant est maintenant celui déplacé.

MOVERENAME

Déplace le fichier au chemin désigné comme argument en renommant le fichier. Le chemin obtenu sera un chemin absolu (et non un chemin relatif).

- Une fois le Path transformé selon les remplacements dynamiques, il est utilisé comme String Format avec le Transfer Information utilisé en entrée (String.format(Path,Info)). Le chemin obtenu doit être un chemin absolu.
- Delay est ignoré.
- Le fichier est marqué comme déplacé.

Exemple :

```
<task>
  <type>MOVERENAME</type>
  <path>/newpath/newfilename</path>
  <delay/>
</task>
```

Le fichier sera déplacé (non copié) dans le répertoire /newpath/ avec comme nouveau nom /newpath/newfilename. Le fichier courant est maintenant celui déplacé.

LINKRENAME

Crée un lien vers le fichier courant et pointe dessus.

- Le lien est d'abord tenté en mode « hard link », puis « soft link » et si ce n'est pas possible (non supporté par le système de fichiers), il crée une copie avec le nouveau nom.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`). Le chemin obtenu doit être un chemin absolu.
- `Delay` est ignoré.
- Le fichier est marqué comme déplacé.

Exemple :

```
<task>
  <type>LINKRENAME</type>
  <path>/newpath/filenamelink</path>
  <delay/>
</task>
```

Le fichier sera un lien dans le répertoire `/newpath/` avec pour nom `filenamelink` (ou une copie si ce n'est pas possible).

RENAME

Renomme le fichier au chemin désigné comme argument. Le chemin obtenu sera un chemin absolu (et non un chemin relatif).

- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`). Le chemin obtenu doit être un chemin absolu.
- `Delay` est ignoré.
- Le fichier est marqué comme déplacé.

Exemple :

```
<task>
  <type>RENAME</type>
  <path>/newpath/newfilename</path>
  <delay/>
</task>
```

Le fichier sera déplacé avec le nouveau nom spécifié. Le fichier est marqué comme déplacé.

DELETE

Cette tâche efface le fichier courant.

- Le fichier courant n'est plus valide.
- Aucun autre argument n'est pris en compte.

Exemple :

```
<task>
  <type>DELETE</type>
  <path/>
  <delay/>
</task>
```

Le fichier courant est effacé. En conséquence, plus aucune action ne peut être opérée sur le fichier. Note : si le fichier ne peut pas être effacé, un Warning sera levé.

VALIDFILEPATH

Teste si le fichier courant est sous l'un des dossiers obtenus depuis le `Path` ou les `Transfer Information`.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Le résultat devra être : `path1 path2 ...` où chaque chemin est séparé par un « blanc ».
- Si `Delay` n'est pas 0, un log sera produit.
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>VALIDFILEPATH</type>
  <path>/path1/ /path2/</path>
  <delay>1</delay>
</task>
```

Ceci vérifiera si le fichier courant est dans un des dossiers spécifiés, ici `/path1` ou `/path2`. Et il fera une sortie log pour enregistrer le résultat de cette vérification.

6.5.4 Tâches agissant sur le fichier

TAR

Crée un TAR depuis les arguments comme source et destination ou un UNTAR des fichiers depuis une archive TAR.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Si `delay` est `1`, l'archive tar indiquée en premier argument est **extraite** dans le dossier indiqué en second argument (le `path archiveFile destDir` équivaut à la commande `tar xf archiveFile -C destDir`)
- Si `delay` est `2`, l'archive tar indiquée en premier argument est **crée** avec le contenu du dossier indiqué en second argument (le `path archiveFile sourceDir` équivaut à la commande `tar cf archiveFile sourceDir`)
- Si `delay` est `3`, l'archive tar indiquée en premier argument est **crée** avec les fichiers indiqués dans les arguments suivants (le `path archiveFile sourceFile1 sourceFile2` équivaut à la commande `tar cf archiveFile sourceFile1 sourceFile2`)
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>TAR</type>
  <path>/path/sourcetarfile /path/targetdirectory/</path>
  <delay>1</delay>
</task>
```

Ceci déclenchera un `UNTAR` depuis l'archive TAR `/path/sourcetarfile` vers le dossier `/path/targetdirectory`. Le fichier n'est pas marqué comme déplacé.

ZIP

Crée un ZIP depuis les arguments comme source et destination ou un UNZIP des fichiers depuis une archive ZIP.

- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Si `delay` est `1`, l'archive zip indiquée en premier argument est **extraite** dans le dossier indiqué en second argument (le `path archiveFile destDir` équivaut à la commande `unzip archiveFile -d destDir`)
- Si `delay` est `2`, l'archive zip indiquée en premier argument est **crée** avec le contenu du dossier indiqué en second argument (le `path archiveFile sourceDir` équivaut à la commande `zip -r archiveFile sourceDir`)
- Si `delay` est `3`, l'archive zip indiquée en premier argument est **crée** avec les fichiers indiqués dans les arguments suivants (le `path archiveFile sourceFile1 sourceFile2` équivaut à la commande `zip archiveFile sourceFile1 sourceFile2`)
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>ZIP</type>
  <path>/path/sourcetarfile /path/targetdirectory/</path>
  <delay>1</delay>
</task>
```

Ceci déclenchera un `UNZIP` depuis l'archive ZIP `/path/sourcetarfile` vers le dossier `/path/targetdirectory`. Le fichier n'est pas marqué comme déplacé.

COMPRESS

Nouveau dans la version 3.6.0 : La tâche de compression unitaire efficace et performante est ajoutée.

Crée un fichier compressé au format ZSTD depuis le fichier courant comme source et comme destination si spécifié le nom du fichier, sinon le nom du fichier existant avec l'extension `.zstd`, ou inversement décompresse le fichier courant vers le om du fichier spécifié, et si non spécifié le nom du fichier courant existant avec l'extension `.unzstd`.

- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Si `delay` est `1`, il s'agit d'une décompression
- Si `delay` est `0`, il s'agit d'une compression
- Le fichier est marqué comme déplacé et modifié pour la nouvelle cible.

Exemple :

```
<task>
  <type>COMPRESS</type>
  <path>/path/targetdirectory/newFile.zstd</path>
  <delay>0</delay>
</task>
```

Ceci déclenchera une compression au format ZSTD du fichier courant vers vers le fichier `/path/targetdirectory/newFile.zstd`. Le fichier est marqué comme déplacé avec ce nouveau fichier.

TRANSCODE

Permet de transcoder un fichier d'un ensemble de codage vers un autre.

Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).

- `-from fromCharset`
- `-to toCharset`
- `-newfile filename` argument optionnel : si non utilisé, ce sera le nom du fichier courant plus `.extension` (usuellement `transcode`) ; si utilisé, aucune extension ne sera ajoutée
- `-extension extension` argument optionnel : si non utilisé, le fichier produit sera `filename.transcode`
- `-dos2unix` or `-unix2dos` argument optionnel, mais si présent, `-from` et `-to` peuvent être ignorés ; ceci autorise des actions `dos2unix/unix2dos` à la fin du transcodage. Cette opération peut être réalisée même sans les options `-from` ou `-to`, ce qui signifie que seule cette transformation sera appliquée, sans transcodage.

`fromCharset` et `toCharset` sont des chaînes représentant les codages officiels disponibles en Java dont.

Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>TRANSCODE</type>
  <path>-from fromCharset -to toCharset -newfile /path/file</path>
  <delay/>
</task>
```

Ceci transcodera le fichier courant depuis `fromCharset` vers `toCharset` et le résultat sera placé dans le fichier `/path/file`. Le fichier n'est pas marqué comme déplacé.

Une méthode en ligne de commande (depuis Waarp Common) permet d'obtenir une liste en html (`-html`), csv (`-csv`) ou au format texte (`-text`) de tous les codages supportés par votre JVM. Pour l'utiliser, exécuter la commande suivante :

```
java -cp WaarpCommon-1.2.7.jar \
  org.waarp.common.transcode.CharsetsUtil \
  [-csv | -html | -text ]
```

Elle peut également être utilisé pour transcoder des fichiers en dehors de R66.

```
java -cp WaarpCommon-1.2.7.jar \
  org.waarp.common.transcode.CharsetsUtil \
  -from fromFilename fromCharset -to toFilename toCharset
```

Codages supportés

Parmi les ensembles de codages, les plus connus sont :

- France : IBM297 or IBM01147
- Italy : IBM280 or IBM01144
- UK : IBM285 or IBM01146
- International (Switzerland, Belgium) : IBM500 or IBM01148
- Austria/Germany : IBM273 or IBM01141
- Spain and Latin America : IBM284 or IBM01145
- Portugal, Brazil, USA, Canada, Netherlands : IBM037 or IBM01140
- Central and Eastern Europe : IBM870
- Cyrillic : x-IBM1025 (x-IBM1381 ?)
- Turkey : IBM1026

- Cyrillic Ukraine : x-IBM1123
- Denmark, Norway : IBM277 or IBM01142
- Finland or Sweden : IBM278 or IBM01143
- Greece : x-IBM875 or x-IBM1124

Voir aussi :

- référence IBM des code pages

UNZEROED

Cette tâche ajoute un octet à un fichier si celui-ci est vide (de taille 0).

Cette tâche sera en erreur si le fichier est de taille 0 mais ne peut pas être « unzeroed ». Si le chemin est non vide, le contenu sera utilisé comme remplissage du le fichier vide. S'il est vide, le caractère « blanc » sera utilisé.

Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).

- Si `Delay` est 1, la tâche produira un log de niveau info
- Si `Delay` est 2, la tâche produira un log de niveau warn
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>UNZEROED</type>
  <path>optional</path>
  <delay>1</delay>
</task>
```

Ceci remplira le fichier courant s'il est vide avec le contenu « optional » et produiera un log de niveau INFO en l'absence d'erreur, de niveau ERROR en cas d'erreur.

CHKFILE

Cette tâche vérifie différentes propriétés relatives au fichier courant en fonction des arguments.

Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).

- `SIZE LT/GT/LTE/GTE/EQ number`
  - vérifie la taille du fichier en fonction d'une limite (plus petit, plus grand, plus petit ou égal, plus grand ou égal, égal)
- `DFCHECK`
  - vérifie que la taille du fichier à recevoir est compatible avec l'espace disponible restant tant sur l'espace de travail que sur l'espace final de réception (depuis le contexte)
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>CHKFILE</type>
  <path>SIZE LT 1000000 SIZE GT 1000 DFCHECK</path>
  <delay/>
</task>
```

Ceci testera si le fichier est plus petit que 10 MO (base 10), plus grand que 1000 octets et si les répertoires de travail et de réceptions ont assez d'espace pour y écrire le fichier (taille annoncée par l'émetteur).

CHMOD

Cette tâche permet de modifier les droits du fichier (comme la commande `CHMOD` sous Unix) avec les arguments suivants :

- le chemin complet est celui du fichier courant
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- les arguments in fine seront de la forme `[ua][+-=][rwx]` où de multiples répétitions peuvent être spécifiées, séparées par un caractère `blanc`
  - `u/a` signifiant l'utilisateur (Utilisateur système Waarp)/all (tous) (groupe et autre n'existent pas en Java),
  - `+/-/=` signifiant l'ajout, le retrait ou l'affectation (l'affectation signifie que tous les autres droits sont retirés),
  - `r/w/x` signifiant Read/Write/Execute (Lecture/Ecriture/Exécution)
- Le fichier n'est pas marqué comme déplacé.

Par exemple :

- `u=rwx a=r`
- `ua+rw`
- `u=rw a-wx`
- `a+rw`

Si plusieurs modes sont indiqués, ils seront exécutés en séquence. Ainsi `a=r a+w a-r` donnera `a=w`.

Exemple :

```
<task>
  <type>CHMOD</type>
  <path>a=r a+w a-r</path>
  <delay/>
</task>
```

ICAP

Nouveau dans la version 3.4.0.

Voir aussi :

Une documentation complète d'installation au regard des interactions avec un serveur ICAP est disponible *ici*

Cette tâche permet l'échange avec un serveur répondant à la norme RFC 3507 dite *ICAP*. Elle permet de transférer le contenu du fichier vers un service ICAP via une commande *RESPMOD* et d'obtenir la validation de ce fichier par le service (status *204*).

La liste des arguments est la suivante :

- `-file filename` spécifie le chemin du fichier sur lequel opérer (si le nom est `EICARTEST`, un faux virus de test basé sur EICAR test sera envoyé).
- `-to hostname` spécifie l'adresse (via DNS ou IP) du serveur ICAP
- [`-port port`, défaut 1344] spécifie le port à utiliser (défaut 1344)
- `-service name` | `-model name` spécifie le service ou modèle ICAP à utiliser
- [`-previewSize size`, défaut aucun] spécifie la taille de Preview à utiliser (défaut négociée)
- [`-blockSize size`, défaut 8192] spécifie la taille en émission à utiliser (défaut 8192)
- [`-receiveSize size`, défaut 65536] spécifie la taille en réception à utiliser (défaut 65536)
- [`-maxSize size`, défaut MAX_INTEGER] spécifie la taille maxmale d'un fichier à utiliser (défaut MAX_INTEGER)
- [`-timeout in_ms`, défaut equiv à 10 min] spécifie la limite de temps à utiliser (défaut equiv à 10 min)
- [`-keyPreview key -stringPreview string`, défaut aucun] spécifie la clef et la chaîne associée pour Options à valider (défaut aucun)

— [-key204 key -string204 string, défaut aucun] spécifie la clef et la chaîne associée pour 204 ICAP à valider (défaut aucun)
— [-key200 key -string200 string, défaut aucun] spécifie la clef et la chaîne associée pour 200 ICAP à valider (défaut aucun)
— [-stringHttp string, défaut aucun] spécifie la chaîne pour HTTP 200 ICAP à valider (défaut aucun)
— [-logger DEBUG|INFO|WARN|ERROR, défaut aucun] spécifie le niveau de log entre DEBUG | INFO | WARN | ERROR (défaut WARN)
— [-errorMove path | -errorDelete | -sendOnError] spécifie l'action en cas de scan erronné : un répertoire de quarantaine, l'effacement du fichier, la retransmission (R66) vers un autre partenaire (mutuellement exclusif) (défaut aucun)
— [-ignoreNetworkError] spécifie que sur une erreur réseau, le fichier sera considéré comme OK
— [-ignoreTooBigFileError] spécifie que sur une erreur de fichier trop grand, le fichier sera considéré comme OK

Si une commande R66 de retransfert est demandée (-sendOnError), la dernière option pour ICAP devra être suivie de -- avant de poursuivre sur les options usuelles pour la commande TRANSFER.

Exemple 1 :

```
<task>
  <type>ICAP</type>
  <path>-file #TRUEFULLPATH# -to hostname -service name
  -previewSize size -blockSize size -receiveSize size
  -maxSize size -timeout in_ms
  -keyPreview key -stringPreview string
  -key204 key -string204 string
  -key200 key -string200 string
  -stringHttp string -logger WARN -errorDelete
  -ignoreNetworkError</path>
  <delay>10000</delay>
</task>
```

Ici, en cas de scan en erreur, le fichier sera effacé.

Exemple 2 :

```
<task>
  <type>ICAP</type>
  <path>-file #TRUEFULLPATH# -to hostname -model name
  -previewSize size -blockSize size -receiveSize size
  -maxSize size -timeout in_ms
  -keyPreview key -stringPreview string
  -key204 key -string204 string
  -key200 key -string200 string
  -stringHttp string -logger WARN -errorMove path
  -ignoreNetworkError</path>
  <delay>10000</delay>
</task>
```

Ici, en cas de scan en erreur, le fichier sera déplacé dans un autre répertoire.

Exemple 3 :

```
<task>
  <type>ICAP</type>
  <path>-file #TRUEFULLPATH# -to hostname -model name
```

(suite sur la page suivante)

(suite de la page précédente)

```
  -previewSize size -blockSize size -receiveSize size
  -maxSize size -timeout in_ms
  -keyPreview key -stringPreview string
  -key204 key -string204 string
  -key200 key -string200 string
  -stringHttp string -logger WARN -sendOnError
  -ignoreNetworkError -- -file #TRUEFULLPATH# -to
  requestedHost -rule rule [-copyinfo]
  [-info information]</path>
  <delay>10000</delay>
</task>
```

Ici, en cas de scan en erreur, le fichier sera envoyé vers un autre serveur (l'effacement sera alors pris en charge par la règle utilisée pour l'envoyer).

Exemple 4 :

```
<task>
  <type>ICAP</type>
  <path>-file #TRUEFULLPATH# -to hostname -model ICAP_AVSCAN
  -sendOnError -ignoreNetworkError -ignoreTooBigFileError --
  -file #TRUEFULLPATH# -to requestedHost -rule rule -copyinfo
  -info FILE INFECTED</path>
  <delay>10000</delay>
</task>
```

Même cas que l'exemple 3 plus minimaliste et réaliste.

6.5.5 Tâches exécutant un sous-traitement

EXEC

Exécute une commande externe en fonction des arguments `Path` et `Transfer Information`.

- Le `Delay` est le temps maximum autorisé en millisecondes avant que la tâche ne soit considérée comme en time out et donc en erreur.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Le fichier n'est pas marqué comme déplacé.

La commande externe est supposée se comporter comme suit pour ses valeurs de retour :

- exit 0, pour une exécution correcte
- exit 1, pour une exécution correcte mais avec avertissement
- toute autre valeur pour une exécution en erreur

Exemple :

```
<task>
  <type>EXEC</type>
  <path>/path/command arguments #TRANSFERID# #TRUEFULLPATH# %s</path>
  <delay>10000</delay>
</task>
```

En prenant en compte les transformations dynamiques, la commande `/path/command` sera exécutée avec les arguments suivants : `arguments transferId /path/currentFilename transferInformation`.

EXECMOVE

Exécute une commande externe en fonction des arguments `Path` et `Transfer Information`.

- Le `Delay` est le temps maximum autorisé en millisecondes avant que la tâche ne soit considérée comme en time out et donc en erreur.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- La dernière ligne retournée par la commande externe est interprétée comme le nouveau chemin absolu du fichier courant. La commande externe est responsable d'avoir réellement déplacer le fichier vers ce nouvel emplacement.
- Le fichier est marqué comme déplacé.

La commande externe est supposée se comporter comme suit pour ses valeurs de retour :

- exit 0, pour une exécution correcte
- exit 1, pour une exécution correcte mais avec avertissement
- toute autre valeur pour une exécution en erreur

Exemple :

```
<task>
  <type>EXECMOVE</type>
  <path>/path/command arguments #TRANSFERID# #TRUEFULLPATH# %s</path>
  <delay>10000</delay>
</task>
```

En prenant en compte les transformations dynamiques, la commande `/path/command` sera exécutée avec les arguments suivants : `arguments transferId /path/currentFilename transferInformation`. La dernière ligne retournée par la commande externe est interprétée comme le nouveau chemin absolu du fichier courant.

EXECOUTPUT

Exécute une commande externe en fonction des arguments `Path` et `Transfer Information`.

- Le `Delay` est le temps maximum autorisé en millisecondes avant que la tâche ne soit considérée comme en time out et donc en erreur.
- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Toutes les lignes retournées par la commande externe (sortie standard) sont interprétées comme un possible message d'erreur.
- Le fichier n'est pas marqué comme déplacé, sauf en cas d'erreur et si `NEWFILENAME` est utilisé comme préfixe au nom du fichier).

La commande externe est supposée se comporter comme suit pour ses valeurs de retour :

- exit 0, pour une exécution correcte
- exit 1, pour une exécution correcte mais avec avertissement
- toute autre valeur pour une exécution en erreur et seulement dans ce cas, la sortie standard est utilisée comme message d'erreur. Des informations peuvent être retournées au serveur distant avec les balises `#ERRORMSG#` et `#ERRORCODE#` ou `#ERRORSTRCODE#`, et `NEWFINALNAME` si le fichier a changé.

Exemple :

```
<task>
  <type>EXECOUTPUT</type>
  <path>/path/command arguments #TRANSFERID# #TRUEFULLPATH# %s</path>
  <delay>10000</delay>
</task>
```

En prenant en compte les transformations dynamiques, la commande `/path/command` sera exécutée avec les arguments suivants : `arguments transferId /path/currentFilename transferInformation`.

La dernière ligne retournée par la commande externe est interprétée comme le nouveau chemin absolu du fichier courant. Des informations peuvent être retournées au serveur distant avec les balises #ERRORMSG# et #ERRORCODE# ou #ERRORSTRCODE#, et NEWFINALNAME si le fichier a changé.

EXECJAVA

Exécute une classe Java externe en fonction des arguments Path et Transfer Information.

- Le Delay est le temps maximum autorisé en millisecondes avant que la tâche ne soit considérée comme en time out et donc en erreur.
- Une fois le Path transformé selon les remplacements dynamiques, il est utilisé comme String Format avec le Transfer Information utilisé en entrée (String.format(Path,Info)).
- Le nom de la classe Java (qui doit implémenter R66Runnable ou étendre AbstractExecJavaTask, en ignorant les méthodes validate/finalValidate/invalid utilisées uniquement pour les tâches Business) est obtenu comme le premier argument. L'allocation est réalisée sous la forme new MyClass(), c'est-à-dire un constructeur sans argument.
- Le fichier n'est pas marqué comme déplacé.

Exemple :

```
<task>
  <type>EXECJAVA</type>
  <path>java.class.name #TRANSFERID# #TRUEFULLPATH#</path>
  <delay>10000</delay>
</task>
```

Ceci va déclencher l'exécution de la commande nommée java.class.name avec les arguments suivants : arguments transferId /path/currentFilename.

Eléments additionnels : Usage de la classe ExecJava

Afin de faciliter l'intégration dans des modules applicatifs, Waarp R66 supporte la possibilité de déclencher des classes Java spécifiques de 3 manières (depuis la version 2.3) :

- L'une est au travers de tâches de traitement pré- ou post-transfert, ou en cas d'erreur en utilisant le mot clef EXECJAVA, suivi du nom complet de la classe Java qui doit implémenter l'interface R66Runnable.
- Une autre est d'exécuter des commandes spécifiques R66Business, qui sont également des implémentations de l'interface R66Runnable au travers de l'extension de AbstractExecJavaTask.
- Enfin, il y a la possibilité d'associer une classe Business (voir R66BusinessInterface) au travers d'une « factory » Business (voir R66BusinessFactoryInterface) pour chacun des transfer et qui déclenche différentes méthodes lors des étapes de chaque transfert :
  - void checkAtStartup(R66Session session) : lancé au démarrage avant les tâches de pré-tâches
  - void checkAfterPreCommand(R66Session session) : lancé après les pré-tâches mais avant le transfert
  - void checkAfterTransfer(R66Session session) : lancé après le transfer mais avant les post-tâches
  - void checkAfterPost(R66Session session) : lancé après les post-tâches et avant la fin de la requête
  - void checkAtError(R66Session session) : lancé si une erreur intervient
  - void checkAtChangeFilename(R66Session session) : lancé si le nom du fichier change durant des tâches
  - void releaseResources() : lancé à la toute fin pour nettoyer les possibles ressources utilisées
  - String getInfo() and void setInfo(String info) : lancés de manière programmatique (code métier) pour permettre de positionner une information spéciale (chaîne de caractères) et de la récupérer à n'importe quel moment

Notez que la R66BusinessFactory peut être déclarée dans le fichier XML de configuration du moniteur dans la balise businessfactory dans les parties server ou client, mais est limitée à un constructeur sans argument.

Notez enfin que pour autoriser des requêtes Business, le droit doit avoir été accordé au partenaire comme suit dans le fichier de configuration XML :

```
<business><businessid>hostname</businessid>...</business>
```

Si non positionné, le partenaire ne sera pas autorisé. Pour `EXECJAVA`, la sécurité est assurée par le fait que la règle est locale au serveur qui l'ecécute et que la règle peut elle aussi limiter les partenaires qui peuvent l'utiliser.

RESTART

Cette tâche permet de redémarrer un serveur Waarp. Il n'y a aucun argument.

Exemple :

```
<task>
  <type>RESTART</type>
  <path></path>
  <delay>0</delay>
</task>
```

L'exemple d'usage le plus fréquent est la mise à jour des binaires ou de la configuration XML du serveur via un transfert, suivi d'un `UNTAR` ou `UNZIP` et enfin d'un `RESTART`.

6.5.6 Tâches exécutant un transfert

TRANSFER

Nouveau dans la version 3.4.0 : option `-nofollow`

Soumet un nouveau transfert basé sur des arguments `Path` et `Transfer Information`.

- Une fois le `Path` transformé selon les remplacements dynamiques, il est utilisé comme `String Format` avec le `Transfer Information` utilisé en entrée (`String.format(Path,Info)`).
- Les arguments de transferts sont obtenus à partir du `Path` transformé.
- Le résultat est considéré comme un `r66send` sauf `-info` qui doit être le dernier item, et `-copyinfo` copiera en première position les informations de transferts originales dans les nouvelles, en ayant toujours la possibilité d'en ajouter d'autres via `-info`
- `Delay` est ignoré
- Le fichier n'est pas marqué comme déplacé.

Arguments du transfert :

```
-to <arg>         Spécifie le partenaire distant
(-id <arg>|       Spécifie l'identifiant du transfert
 (-file <arg>     Spécifie le fichier à opérer
  -rule <arg>))   Spécifie la règle de transfert
[-block <arg>]    Spécifie la taille du bloc
[-nofollow]       Spécifie que le trasfert ne devra pas intégrer un "follow" id
[-md5]            Spécifie qu'un calcul d'empreinte doit être réalisé pour
                   valider le transfert
[-delay <arg>|    Spécifie le délai comme un temps epoch ou un délai (+arg) en ms
 -start <arg>]    Spécifie la date de démarrage yyyyMMddHHmmss
[-nolog]          Spécifie de ne rien conserver de ce transfert (en base)
[-notlogWarn |    Spécifie que le log final est en mode Info si OK
 -logWarn]        Spécifie que le log final est en mode Warn si OK (défaut)
[-copyinfo]       Spécifie que les informations de transfert seront recopiées
                   intégralement en préposition des nouvelles valeurs
[-info <arg>)     Spécifie les informations de transfert (en dernière position)
```

Exemple :

```
<task>
  <type>TRANSFER</type>
  <path>-file #TRUEFULLPATH# -to remotehost
  -rule ruletouse -info transfer Information</path>
  <delay/>
</task>
```

Ceci créera une nouvelle requête de transfert (asynchrone) en utilisant le fichier courant (`#TRUEFULLPATH#`), pour envoyer (ou recevoir selon la règle utilisée) vers (ou depuis) le partenaire, en utilisant `transfer Information` comme argument de transfert.

RESCHEDULE

Replanifie une tâche de transfert en cas d'erreur avec un délai spécifié en millisecondes, si le code d'erreur est un de ceux spécifiés et si les intervalles optionnels de dates sont compatibles avec la nouvelle planification.

La balise `path` accepte les arguments suivants (les deux premiers sont obligatoires) :

- `-delay ms` spécifie le délai en millisecondes après lequel retenter ce transfert
- `-case errorCode,errorCode,...` où les « errorCode » sont une liste de codes d'erreur pour lesquels la tâche est exécutée. Les codes suivants sont disponibles (e nom de l'erreur et le code d'une lettre peuvent petre utilisés) :
  - `ConnectionImpossible(C)`,
  - `ServerOverloaded(l)`
  - `BadAuthent(A)`,
  - `ExternalOp(E)`
  - `TransferError(T)`
  - `MD5Error(M)`
  - `Disconnection(D)`
  - `RemoteShutdown(r)`
  - `FinalOp(F)`
  - `Unimplemented(U)`
  - `Shutdown(S)`
  - `RemoteError(R)`
  - `Internal(I)`
  - `StoppedTransfer(H)`
  - `CanceledTransfer(K)`
  - `Warning(W)`
  - `Unknown(-)`
  - `QueryAlreadyFinished(Q)`
  - `QueryStillRunning(s)`
  - `NotKnownHost(N)`,
  - `QueryRemotelyUnknown(u)`
  - `FileNotFound(f)`
  - `CommandNotFound(c)`
  - `PassThroughMode(p)`
- `-between starttime;endtime`, `-notbetween starttime;endtime` permettent de définir des plages horaires durant lesquelles les tentatives de transferts peuvent ou, respectivement, ne peuvent pas être retentés. Les règles suivantes sont utilisées :
  - Ces arguments peuvent être utilisés plusieurs fois et peuvent être mixés ;
  - Ils ont le format suivant : `Yn:Mn:Dn:Hn:mn:Sn` où n spécifie un nombre pour chaque partie d'une date (optionnelle) comme `Y` = Année, `M` = Mois, `D` = Jour, `H` = Heure, `m` = minute, `S` = seconde ;

  - Le format peut être `X+n`, `X-n`, `X=n` ou `Xn` où `X+-n` signifie ajouter/soustraire n à la date courante, tandis que `X=n` ou `Xn` signifie une valeur exacte ;
  - Si aucune spécification de temps n'est présente, ce sera la date actuelle ;
  - La date planifiée ne doit pas être dans un des intervalles définis par les arguments `-notbetween` ;
  - La date planifiée doit être dans un des intervalles définis par les arguments `-between` ;
  - Si aucun de ces arguments n'est spécifié, la date planifiée sera toujours valide.
  - Si `starttime` est plus grand que `endtime`, `endtime` prendra la valeur `starttime` + 1 jour ;
  - Si `starttime` et `endtime` sont inférieurs à la date planifiée, ils auront également un décalage d'un jour.
- `-count limit` sera la limite de retentatives. La valeur limite est prise des `information de transfert` et non de la règle.
  - Chaque fois que cette fonction est appelée, la valeur limite est remplacée par `newlimit = limit - 1` dans l'`information de transfert`.
  - Pour assurer la cohérence, la valeur doit être dans ce champ puisque elle sera changée statiquement. Cependant, une valeur doit être positionnée dans la règle afin de réinitialiser la valeur lorsque le décompte tombe à 0.
  - Ainsi, dans la règle, `-count resetlimit` doit être présent, où `resetlimit` sera la nouvelle valeur lorsque celle-ci atteindra 0. Si elle est manquante la condition ne peut pas être appliquée.

Important :

- Notez que si un précédent appel à `RESCHEDULE` a été réalisé et courroné de succès, les appels suivants seront ignorés.
- Toutes tâches qui suivent celle-ci seront ignorées et non exécutées si la replanification est acceptée. Au contraire, si la replanification est refusée, les tâches suivantes seront exécutées normalement.

Exemple :

```
<task>
  <type>RESCHEDULE</type>
  <path>-delay 3600000
  -case ConnectionImpossible,ServerOverloaded,Shutdown
  -notbetween H7:m0:S0;H19:m0:S0
  -notbetween H1:m0:S0;H=3:m0:S0 -count 1</path>
  <delay/>
</task>
```

Cet exemple illustre le cas d'une nouvelle tentative d'un transfert tombé en erreur à cause d'une connexion impossible. La nouvelle tentative sera faite dans une heure, si l'heure résultante n'est pas comprise 7H du matin et 7H du soir, ni entre 1H du matin et 3H du matin avec une limite de 3 tentatives (la valeur `retry` sera réinitialisée à 1 en cas de 3 tentatives).

Pour chaque tentative, le compteur sera décrémenté.

FTP

Cette tâche permet de réaliser un transfert synchrone en utilisant FTP. Elle utilise les paramètres suivants :

- `-file filepath`
- `-to requestedHost`
- `-port port`
- `-user user`
- `-pwd pwd`
- `[-account account]`
- `[-mode active/passive]`
- `[-ssl no/implicit/explicit]`

— [-cwd remotepath]
— [-digest (crc,md5,sha1)]
— [-pre extraCommand1 avec "," comme séparateur d'arguments]
— -command command où commande est un parmi (get, put, append)
— [-post extraCommand2 avec "," comme séparateur d'arguments]

L'orde des commandes sera alors :

1. Connexion au requestHost avec le port.

   Si -ssl vaut implicit, une liaison liaison TLS native est utilisée et l'étape 5 n'est pas exécutée

2. USER user
3. PASS pwd
4. ACCT account, si -account est renseigné
5. AUTH TLS, PBSZ 0 et PROT P, si -ssl vaut explicit
6. PASV, si -mode vaut passive
7. CWD remotepath

   En cas d'erreur, le dossier est créé : MKD remotepath puis CWD remotepath (en ignorant les erreurs)

8. Si -pre est renseigné, extraCommand1 avec "," remplacés par " "

   note : n'utilisez pas des commande standards FTP comme ACCT, PASS, REIN, USER, APPE, STOR, STOU, RETR, RMD, RNFR, RNTO, ABOR, CWD, CDUP, MODE, PASV, PORT, STRU, TYPE, MDTM, MLSD, MLST, SIZE, AUTH

9. BINARY (binary format)
10. Transfert des données :

— Si -command vaut get, RETR filepath.basename
— Si -command vaut put`, ``STOR filepath
— Si -command vaut append, APPE filepath.basename

11. Si l'argument -digest est donné et que le serveur FTP distant est compatible avec les commandes XCRC, XMD5, XSHA1, FEAT (le résultat vérifie la présente des options disponibles) ; puis XCRC/XMD5/XSHA1 filepath.basename ; puis localement il y aura la comparaison de ce hash avec le fichier local
12. Si -post est renseigné, extraCommand2 avec "," remplacés by " "

    note : n'utilisez pas des commande standards FTP comme ACCT, PASS, REIN, USER, APPE, STOR, STOU, RETR, RMD, RNFR, RNTO, ABOR, CWD, CDUP, MODE, PASV, PORT, STRU, TYPE, MDTM, MLSD, MLST, SIZE, AUTH

13. QUIT

Le fichier courant est inchangé et non marqué comme déplacé.

Exemple :

```
<task>
  <type>FTP</type>
  <path>-file /path/file -to remotehost -port port
  -user username -pwd password -command put</path>
  <delay/>
</task>
```

Ceci enverra (put) le fichier /path/file au serveur FTP remotehost sur le port port en utilisant les username et password.

6.5.7 Tâches agissant sur l'emplacement du fichier via un stockage S3

Nouveau dans la version 3.6.0 : Les tâches ci-dessous sont nouvelles et supportées avec le module WaarpR66-S3.

S3GET

Copie le fichier depuis un stockage S3 Objet comme argument et remplace le fichier actuel avec ce fichier comme source.

- `Delay` est ignoré.
- La commande finale est fonction des arguments `Path` et `Transfer Information`.
- Le fichier est marqué comme déplacé.

La règle à utiliser doit être en mode THROUGHMODE, soit `SENDMD5THROUGHMODE` ou `SENDTHROUGHMODE` (respectivement 7 ou 5) en mode `SEND`, soit `RECVMD5THROUGHMODE` ou `RECVTHROUGHMODE` (respectivement 8 ou 6) en mdoe `RECV`, car le fichier n'existe pas au démarrage.

Le format est le suivant :

- `-URL url` du service S3
- `-accessKey access Key` du service S3
- `-secretKey secret Key` du service S3
- `-bucketName bucket Name` où est stocké l'objet
- `-sourceName source Name` dans le bucket pour sélectionner l'objet final
- `-file final File path` absolue ou relatif depuis le chemin IN
- [`-getTags` [* or `liste` de noms de tag séparés par des virgules sans espace]]

Les actions seront dans l'ordre :

1) connexion au service S3 en utilisant la clef d'accès et la clef de secret
2) Récupère depuis le bucket l'objet source et le stocke dans le fichier spécifié
3) Si getTags is positionné, les informations sont ajoutées au transferInfo et fileInfo
4) le fichier courrant est positionné sur ce nouveau fichier reçu (équivalent à la tâche R66 `RENAME`)
5) l'émetteur envoie une mise à jour (nom et taille)

Exemple :

```
<tasks>
  <task>
    <type>S3GET</type>
    <path>-URL %s -accessKey %s -secretKey %s -bucketName %s -sourceName %s -file
↪#TRUEFULLPATH# -getTags key1,key2</path>
    <delay/>
    <rank>0</rank>
  </task>
</tasks>
```

S3GETDELETE

Copie le fichier depuis un stockage S3 Objet comme argument et remplace le fichier actuel avec ce fichier comme source et efface l'objet source.

- `Delay` est ignoré.
- La commande finale est fonction des arguments `Path` et `Transfer Information`.
- Le fichier est marqué comme déplacé.

La règle à utiliser doit être en mode THROUGHMODE, soit `SENDMD5THROUGHMODE` ou `SENDTHROUGHMODE` (respectivement 7 ou 5) en mode `SEND`, soit `RECVMD5THROUGHMODE` ou `RECVTHROUGHMODE` (respectivement 8 ou 6) en mdoe `RECV`, car le fichier n'existe pas au démarrage.

Le format est le suivant :
- `-URL url` du service S3
- `-accessKey access Key` du service S3
- `-secretKey secret Key` du service S3
- `-bucketName bucket Name` où est stocké l'objet
- `-sourceName source Name` dans le bucket pour sélectionner l'objet final
- `-file final File path` absolue ou relatif depuis le chemin IN
- [`-getTags` [* or `liste` de noms de tag séparés par des virgules sans espace]]

Les actions seront dans l'ordre :

1) connexion au service S3 en utilisant la clef d'accès et la clef de secret
2) Récupère depuis le bucket l'objet source et le stocke dans le fichier spécifié
3) Si getTags is positionné, les informations sont ajoutées au transferInfo et fileInfo
4) le fichier courrant est positionné sur ce nouveau fichier reçu (équivalent à la tâche R66 `RENAME`)
5) l'émetteur envoie une mise à jour (nom et taille)
6) l'objet S3 est effacé

Exemple :

```
<tasks>
  <task>
    <type>S3GETDELETE</type>
    <path>-URL %s -accessKey %s -secretKey %s -bucketName %s -sourceName %s -file
↪#TRUEFULLPATH# -getTags key1,key2</path>
    <delay/>
    <rank>0</rank>
  </task>
</tasks>
```

S3DELETE

Efface l'objet S3.
- `Delay` est ignoré.
- La commande finale est fonction des arguments `Path` et `Transfer Information`.
- Le fichier courant n'est pas modifié.

Le format est le suivant :
- `-URL url` du service S3
- `-accessKey access Key` du service S3
- `-secretKey secret Key` du service S3
- `-bucketName bucket Name` où est stocké l'objet
- `-sourceName source Name` dans le bucket pour sélectionner l'objet final

Les actions seront dans l'ordre :

1) connexion au service S3 en utilisant la clef d'accès et la clef de secret
2) l'objet S3 est effacé

Exemple :

```
<tasks>
  <task>
    <type>S3DELETE</type>
    <path>-URL %s -accessKey %s -secretKey %s -bucketName %s -sourceName %s</path>
    <delay/>
    <rank>0</rank>
```

(suite sur la page suivante)

(suite de la page précédente)

```
  </task>
</tasks>
```

S3PUT

Copie le fichier courant ver un stockage S3 Objet.
- Delay est ignoré.
- La commande finale est fonction des arguments Path et Transfer Information.
- Le fichier courant est inchangé.

Le format est le suivant :
- -URL url du service S3
- -accessKey access Key du service S3
- -secretKey secret Key du service S3
- -bucketName bucket Name où est stocké l'objet
- -targetName target Name dans le bucket pour sélectionner l'objet final
- [-setTags [clef:valeur,clef:valeur de clef:valeur séparés par des virgules sans espace]]

Les actions seront dans l'ordre :

1) connexion au service S3 en utilisant la clef d'accès et la clef de secret
2) Stocke le fichier courant dans le bucket l'objet destination
3) Si setTags is positionné, les informations sont ajoutées à l'objet S3

Exemple :

```
<tasks>
  <task>
    <type>S3PUT</type>
    <path>-URL %s -accessKey %s -secretKey %s -bucketName %s -targetName %s -setTags␣
↪key1:value1,key2:value2</path>
    <delay/>
    <rank>0</rank>
  </task>
</tasks>
```

S3PUTR66DELETE

Copie le fichier courant ver un stockage S3 Objet et efface le fichier courant.
- Delay est ignoré.
- La commande finale est fonction des arguments Path et Transfer Information.
- Le fichier courant est inchangé.

Le format est le suivant :
- -URL url du service S3
- -accessKey access Key du service S3
- -secretKey secret Key du service S3
- -bucketName bucket Name où est stocké l'objet
- -targetName target Name dans le bucket pour sélectionner l'objet final
- [-setTags [clef:valeur,clef:valeur de clef:valeur séparés par des virgules sans espace]]

Les actions seront dans l'ordre :

1) connexion au service S3 en utilisant la clef d'accès et la clef de secret
2) Stocke le fichier courant dans le bucket l'objet destination
3) Si setTags is positionné, les informations sont ajoutées à l'objet S3

4) le fichier courrant est supprimé (équivalent à la tâche `DELETE`)

Exemple :

```
<tasks>
  <task>
    <type>S3PUTDELETE</type>
    <path>-URL %s -accessKey %s -secretKey %s -bucketName %s -targetName %s -setTags␣
↪key1:value1,key2:value2</path>
    <delay/>
    <rank>0</rank>
  </task>
</tasks>
```

6.6 authent.xml

Le fichier `authent.xml` contient les directives de configurations des partenaires du serveur.

Note : Les changements dans ce fichier sont pris en compte par import dans la base via la commande `waarp-r66server loadconf`.

Les directives de configuration sont réparties en 1 section :
— *identity* : données concernant l'identité d'un partenaire

6.6.1 Section entry

Balise	Type	Obl.	Défaut	Signification
hostid	string	O		Nom de l'hôte
address	string	O		Adresse IP ou nom DNS de l'hôte (si 0.0.0.0, alors c'est un client)
port	integer	O		Port associé à l'adresse pour ce nom
isssl	boolean	N	False	Cette association désigne un hôte usant des connexions chiffrées
admin	boolean	N	False	Cette association désigne un hôte avec droit d'administration via le protocole R66 ; A partir de la version 2.4.9 cela peut être remplacé si par la configuration fichier des rôles
isclient	boolean	N	False	Cette association désigne un hôte de type Client
isactive	boolean	N	True	Vrai si l'entrée est activée/autorisée
isproxified	boolean	N	False	Vrai si l'entrée est accessible via un proxy (désactive la vérification de l'IP)
keyfile	string	N		Fichier contenant la clef publique du partenaire au format GGP
key	string	N		Clef publique du partenaire

`keyfile` ou `key` doivent être spécifiés (l'un des deux uniquement).

CHAPITRE 7

Référence de la base de données

7.1 Référentiels

7.1.1 Référentiels communs

`updatedinfo`

Code	Détail
0	Inconnu
1	Non modifié
2	Interrompu
3	Planifié
4	Erreur
5	En cours
6	Terminé

7.1.2 Table `runner`

`globalstep` et `globallaststep`

Code	Détail
0	Inconnu
1	Traitements pré-transfert
2	Transfert en cours
3	Traitements post-transfert
4	Terminé
5	Erreur

infostatus et stepstatus

Code	Détail
i	Initialistation OK
B	Pré tâches OK
X	Transfert OK
P	Post tâches OK
O	Transfert complet OK
C	Connection en échec
l	Connection en échec pour cause de limitation
A	Mauvaise authentification
E	Opération externe en erreur (pre, post ou erreur)
T	Transfert en erreur
M	Transfert en erreur causé par un problème de hash MD5
D	Déconnexion du partenaire distant
r	Partenaire distant en cours d'arrêt
F	Action finale en erreur
U	Fonctionnalité non implémentée
S	Arrêt en cours
R	Erreur d'execution sur le partenaire distant
I	Erreur interne
H	Arrêt du transfert demandé
K	Annulation du transfert demandé
W	Alerte relevée lors de l'execution
-	Type d'erreur inconnu
Q	Requête déjà terminée par le partenaire distant
s	Requête en cours d'execution
N	Partenaire inconnu
L	Erreur car requête à soit même
u	Requête non trouvée sur le partenaire distant
f	Fichier inconnu
c	Commande inconnue
p	PassThrough demandé mais impossible
z	Traitement en cours
n	Commande erronée
a	Fichier non autorisé
d	Taille de fichier non autorisée

CHAPITRE 8

Liste des changements

La procédure de mise à jour est disponible ici : *Mise à jour*

8.1 Non publié

8.2 Waarp R66 3.6.0 (2021-04-03)

8.2.1 Nouvelles fonctionnalités

- Ajout de l'option `activepassive` pour Gateway FTP avec pour valeur 1 = Active, -1 = Passive, 0 = les deux modes autorisés
- Ajout d'une erreur si le certificat a une date de validité dépassée au démarrage
  - Pas d'erreur d'exécution, il appartient à l'administrateur de gérer les certificats qui sont ingérés dans le keystore ou le truststore
- Waarp R66 : Ajout d'un écran dans l'administrateur Web pour créer ou modifier un Transfert via l'interface Web
  - L'initiateur est le serveur courant
  - Il est possible de définir la règle, le serveur partenaire distant, les informations de transferts (potentiellement vide), le fichier a envoyer (local au serveur et son existence est non testé), optionnellement la date de démarrage (si vide, immédiat), et enfin optionnellement le numéro de suivi (si vide, il sera assigné automatiquement)
  - Il est possible de modifier également un transfert qui n'a pas démarré (toujours avec pour condition le serveur initiateur étant le serveur local)
- Waarp R66 : Ajout d'une possibilité de recherche des transferts associés à un numéro de suivi (en fonction des droits)
- Waarp R66 : Ajout d'une option de logs asynchrones poussés vers une API REST externe ou vers un Elasticsearch (ce dernier uniquement JRE >=8) pour permettre la surveillance globale des transferts de un ou plusieurs moniteurs Waarp R66
- Waarp R66 : Ajout de la possibilité d'étendre les tâches de Waarp R66 via un TaskFactory

- WaarpR66 : Création d'une TaskFactory pour ajouter des tâches R66 qui permettent de lire, écrire ou effacer des fichiers depuis un stockage S3 pouvant servir de source ou cible dans le cas de transferts (org.waarp.openr66.s3.taskfactory.S3TaskFactory) ;
  - Cette Factory est chargée dynamiquement si la classe correspondante est dans le classpath (uniquement disponible en JRE 8 et au-dessus).
- WaarpR66 : Ajout du support de la compression basée sur ZTSD. Cet algorithme est à la fois rapide, peu consommateur et très performant en compression. Il est utilisable via une tâche `COMPRESS` spécifique.
- WaarpR66 : Ajout de la compression à la volée au niveau des blocs (avec l'algorithme ZSTD). Il est activable par la configuration `compression` à `True` dans la partie `limit` du fichier de configuration (active la compression par bloc si le partenaire l'autorise aussi). Cette option est à compléter par l'argument d'information transmis au partenaire avec le mot clef `#COMPRESS#` qui indique qu'il souhaite compresser.
  - Ainsi, si les 2 partenaires ont la compression active et si un transfert précise dans son information de transfert ce mot clef, le transfert utilisera une compression par bloc.
  - Toute autre configuration ne déclenchera pas la compression par bloc (notamment pour compatibilité ascendante).
  - La configuration de compression globale est désactivée par défaut pour réduire la consommation CPU et mémoire, mais elle peut être activée par défaut sans difficulté et n'être jamais utilisée dans des transferts (pas d'argument `#COMPRESS#` ou partenaire n'ayant pas activé cette option), ou n'être utilisée que ponctuellement.
- Compatibilité JRE 16 vérifiée (base JRE11)
- Benchmarks multi-usages et multi-versions (voir chapitre Configuration Avancée)

8.2.2 Correctifs

- Amélioration de la gestion des Threads Clients et Servers
- Amélioration des Threads pour Recv avec minimum/maximum optimisés
- Accroissement de la limite de RUNLIMIT à 50000, maintient du défaut à 1000
- Benchmark sur multiple serveurs Waarp en mode cluster
- Benchmark avec le Monitoring
- Benchmark avec le stockage S3
- Fixe l'usage de Netty Native OpenSSL ou BoringSsl (performances TLS)
- Fixe Waarp R66Proxy
- Fixe les configurations des bases de données, notamment les index et les tailles
- Fixe les vérifications API REST (V1 et V2) (sanity)
- Fixe la transformation Json <-> DbHostAuth
- Fixes de bug liés à la lecture XML des règles, de la gestion de client sans base
- Usage de Saxon pour le standard XML
- Fixes des bugs de stabilités FTP (serveur et clients)
- Benchmark sur Serveur FTP et Gateway FTP (avec H2 et PostgreSQL)
- Amélioration des Types SQL, index et requêtes SQL (R66 principalement)
- Fixe de la gestion des transferts à soi-même
- Ajout d'un test de non régression avec la 3.5.2 qui sert de référence
- Amélioration du code et de la gestion mémoire et de la documentation
- Mise à jour des dépendances, JAR et javascript

8.3 Waarp R66 3.5.2 (2021-03-03)

8.3.1 Correctifs

— Amélioration de la fermeture opportuniste des connexions réseaux
— Amélioration de la liaison JVM/Filesystem lors du test de lisibilité
— Accroissement de la limite de RUNLIMIT à 10000, maintient du défaut à 1000
— Amélioration des performances (usage de ByteBuf natif)
— Fix de la Gateway FTP pour les clients sous Windows qui abusivement envoient une commande OPTS avant l'authentification
— Mise à jour des dépendances

8.4 Waarp R66 3.5.1 (2020-09-01)

8.4.1 Correctifs

— Réduction des temps de latence dans le code
— Correction de mauvaises pratiques de code
— Compatibilité JDK 6 à 11, 3 packages Java (*Waarp*.jar* pour JRE6, *Waarp*-jre8.jar* pour JRE8 et *Waarp*-jre11.jar* pour JRE11)
— Amélioration des performances sensibles (15%) en lien avec les calculs d'empreintes (Digest)
— Amélioration de la documentation
— Amélioration du service Rest V2 pour inclure les informations des FileMonitoring
— Amélioration des logs (performances et level)
— Amélioration des accès base de données (Rule, Host, Business) par cache et optimisation spécifique pour TaskRunner (mise à jour du compteur Rank)
— Mise à jour des dépendances

8.5 Waarp R66 3.5.0 (2020-09-01)

8.5.1 Nouvelles fonctionnalités

— [#74] Les *interfaces réseaux* sont spécifiables en plus du port à utiliser. Plusieurs interfaces sont possibles (séparées par une virgule).

8.5.2 Correctifs

— [#77] Le risque de dépassement de capacité mémoire directe est largement diminué au profit de l'usage du Pool de mémoire Netty au sein de la JVM. Des optimisations majeures en termes d'allocations/désallocations ont également été effectuées.
— [#72] Le commander pouvait être bloqué dans certains cas. (issue [#65])
— [#71] XMLRuleDAO ne prenait pas en compte les règles de transferts (cas d'un client sans base comme le FileMonitor) (issue [#64])
— [#69] Des actions dans le menu Système de l'interface d'administration étaient manquantes. (issue [#63])
— [#70] Un transfert d'un client vers lui-même (self-transfert) provoquait un effacement du transfert. (issue [#62])
— [#68] La page Web admin était cassée avec les map dans le champ Information de transfert. (issue [#61])
— [#67] Les options de sorties (csv, xml, json, property) sont rétablies et dans un format approprié. (issue [#60], issue [#78])

— [#66] EXECOUTPUT provoquait une erreur de mappage de classe (issue [#59])
— Nettoyage du code (styles)
— Mise à jour des dépendances
— Packaging : Modifications des valeurs Xms et Xmx de Java avec les valeurs recommandées.
— Packaging : Correction des commandes de lancement de transfert sous windows
— Packaging : Correction de l'arrêt des filewatchers et des serveurs R66
— Packaging : Les packages .deb dépendent maintenant directement de Java 8 (et non de la JRE par défaut).
— Packaging : Corrections des erreurs « The %1 service is marked as an interactive service. However, the system is configured to not allow interactive services. This service may not function properly. » lors du démarrage des services sur certaines versions de Windows.
— Packaging : par défaut, les logs clients R66 sont écrits dans des fichiers plutôt que sur la sortie standard.

8.6 Waarp R66 3.4.0 (2020-07-17)

8.6.1 Nouvelles fonctionnalités

— [#49] Pour les transferts, une nouvelle fonctionnalité permet de gérer le suivi fin des retransferts (rebonds entre plusieurs serveurs R66). Cette option positionne un champ dans la partie `information de transfert` de la forme suivante : `{"follow": numeroUnique}` pour le premier transfert et les transferts suivants récupèreront ainsi cette information nativement.
Pour les anciennes versions, il est possible de simuler cette option manuellement en spécifiant pour le premier transfert dans le champ `-info` (`information de transfert`) un Json de type `{"follow": numeroUnique}` en attribuant un numéro unique (comme un timestamp).
Cette option est active par défaut. Pour la désactiver, il faut préciser l'option `-nofolow`.
— L'interface REST V2 intègre l'option de recherche par `followId` (`GET /v2/transfers/?followId=number`). `number` étant possiblement un entier long, il est conseillé de le manipuler en chaîne de caractères.
Pour les anciennes versions, il faut requêter tous les transferts et filtrer ensuite sur le champ `transferInformation` selon la présence d'un champ `follow` suivi d'un numéro au format Json.
— [#48] Une nouvelle tâche nommée `ICAP` est créée afin de permettre l'échange avec un serveur répondant à la norme RFC 3507 dite `ICAP`. Elle permet de transférer le contenu du fichier vers un service ICAP via une commande `RESPMOD` et d'obtenir la validation de ce fichier par le service (statut `204`).
— Packaging : ajout de la commande `icaptest` aux scripts `waarp-r66client` pour tester les paramètres ICAP

8.6.2 Évolutions

— [#51] Les valeurs par défaut des limitations de bande passante ont changées : La limitation globale par défaut est maintenant de 100Gbps, et celle par connexion est de 1Gbps (ces valeurs peuvent être ajustées dans les fichiers de configuration).
— [#51] La valeur par défaut de la RAM maximale utilisée par les services WEB et REST a été abaissée à 1Go (au lieu de 4Go) (cette valeur peut être ajustée dans les fichiers de configuration).

8.6.3 Correctifs

— [#50] Le log géré par LogBack génère parfois des logs au démarrage d'information ou de debug qui peuvent être évités (en conservant les Warnings et les Erreurs) via l'ajout dans le fichier de configuration `logback.xml` les paramètres suivants en tête des options :

```
<statusListener
  class="org.waarp.common.logging.PrintOnlyWarningLogbackStatusListener" />
```

— Packaging : les modèles de configuration intègrent le nouveau `StatusListener` dans la configuration des logs
— [#51] Diminution de l'empreinte mémoire pour le cas des clients simples et diminution de la mémoire côté serveur pour les parties Web et REST. (issue [#52])
— [#51] Si aucun argument `-Xms` n'est passé à la JVM lors du démarrage, la valeur par défaut de la JVM s'applique (en général 4Go).
— [#54] Prise en charge correcte du filtrage par expression régulière dans le *file watcher* (il était impossible de filtrer juste sur le nom d'un fichier situé dans un sous-dossier).
— [#57] Certaines commandes ne fonctionnaient plus suite à un bug sur les logs. (issue [#56])
— Mise à jour des dépendances
— Packaging : les scripts `waarp-r66server` utilisaient la configuration client pour certaines sous-commandes
— Packaging : Arrêt des serveurs avec le signal `HUP` plutôt que `INT`

8.7 Waarp R66 3.3.4 (2020-06-02)

8.7.1 Correctifs

— [#31] Corrige la régression sur la sélection d'un transfert à partir de son ID où le nom du serveur local ne prenait pas en compte si le serveur distant était en mode SSL ou pas (régression en 3.0).
— Corrige la documentation (maven site) pour WaarpHttp
— Corrige les dépendences dans les shading jars et les pom
— Corrige l'interface DbHostConfiguration dans le Web Admin
— Corrige la classe HttpWriteCacheEnable
— [#35] Corrige le Web Admin sur les écrans Listing et CancelRestart pour le tri selon le specialId et pour le boutton « Clear »
— [#37] Corrige l'interface RESTV2 pour les accès avec droits non pris en compte
— Nettoyage du code
— Corrige l'intégration de SonarQube avec Maven
— [#38] Corrige l'exemple de la documentation sur l'authentification HMAC
— [#42] Correction de la signature des requêtes dans l'API REST v2
— [#43] Correction de l'authentification HMAC de l'API REST v2
— [#45] Correction d'un bug sur la taille des paquets

8.8 Waarp R66 3.3.3 (2020-05-07)

8.8.1 Correctifs

— [#20] Corrige l'affichage d'un transfert dont la règle n'existe plus dans l'interface d'administration Web Waarp OpenR66 et empêche l'effacement d'une règle tant qu'il existe au moins un transfert qui l'utilise dans sa définition. (issue [#19])
— [#23] Corrige la prise en compte d'un chemin sous Windows avec qui se double en \ (issue [#22])
— [#25] Corrige l'arrêt immédiat du serveur Waarp GW FTP après son démarrage (introduit en 3.1) (issue [#24])

— [#27] Corrige l'absence de connections à la base de données pour l'interface d'administration en mode Responsive (issue [#26])
— [#30] Corrige la régression sur la répétition à l'infini des tentatives de connexion depuis la version 3.1. Le principe de 3 tentatives avant échec est rétabli.
— Corrige les dépendances externes (et le style)

8.9 Waarp R66 3.3.2 (2020-04-21)

8.9.1 Correctifs

— Corrige les tests Rest V1
— Corrige des méthodes manquantes dans le module WaarpHttp
— Mise à jour des dépendances externes (compatibles Java 6)
— Correction de l'API Rest V2 /v2/hostconfig/ qui retourne versionR66 (version du protocole) et versionBin (version du code)
  — La version retournée par l'API V1 n'est plus conforme suite la mise à jour automatique du schéma de la base de données.
— Corrige une fuite mémoire API Rest
— Corrige le cas du blocage d'un client lorsqu'il n'est pas reconnu par un serveur distant

8.10 Waarp R66 3.3.1 (2020-02-17)

8.10.1 Correctifs

— [#13] Corrige l'oubli du module WaarpPassword dans les autres modules dans les packages *jar-with-dependencies* et en crée un pour WaarpPassword ; Met à jour les dépendances pour SonarQube (usage interne)
— [#9] Corrige une régression sur l'API REST v1 introduite dans la version 3.2.0
— [#10] Corrige une régression qui empêche les ports négatifs pour les partenaires introduite dans la version 3.2.0

8.11 Waarp R66 3.3.0 (2020-01-18)

8.11.1 Améliorations

— Ajout des propriétés suivantes à la sortie des commandes `*send` : `specialid`, `finalPath`, `originalPath`, `statusCode`, `ruleid`, `requested`, `requester`, `fileInformation`, `originalSize`
— Amélioration de la prise en compte d'un transfert échoué sur connexion impossible pour rejeu
— Amélioration de la détection au plus tôt de l'absence d'un fichier lors d'une demande d'émission
— Amélioration de la prise en compte d'un fichier déjà pris en compte par FileWatcher mais modifié après, sans être effacé (ce qui n'est pas une bonne pratique) : le fichier sera reprogrammé pour un nouveau transfert. Cette amélioration est désactivable avec l'option `-ignoreAlreadyUsed=true`
— Mise à jour des dépendances externes

8.12 Waarp R66 3.2.0 (2019-10-25)

8.12.1 Sécurité

— Support de TLS 1.2 pour toutes les versions de JRE

8.12.2 Nouveautés

— Refonte Db

8.12.3 Améliorations

— Diminution du nombre de threads utilisés
— Optimisation de l'utilisation de ressources externes (RAM, CPU)
— Mise à jour des dépendances externes

8.12.4 Correctifs

— Suppressions d'erreurs de type « deadlocks »

8.13 Waarp R66 3.1.0-1 (non publiée)

Note : En raison de bugs bloquants, cette version n'a pas été publiée.

8.13.1 Sécurité

— Corrige un bug permettant de contourner l'obligation d'un canal SSL

8.13.2 Nouveautés

— Nouvelle version de l'API REST ([documentation](interface/restv2/index.html))

8.13.3 Améliorations

— Les regexes du filewatcher permettent de filtrer sur le chemin complet des fichiers et non juste le nom du fichier
— les scripts `waarp-r66client` et `waarp-r66server` permettent de mettre à jour la base de données.

8.13.4 Correctifs

— Corrige les code retour d'initialisation de la base de données
— Corrige les messages d'erreur suite à un échec de connexion
— Renomme l'option `dbcheck` de la configuration de base données en `autoupgrade`
— Corrige les messages d'erreur au chargement de la page « Cancel-Restart » de l'interface d'admin
— Les services sont arrêtés avec le signal `interrupt` plutôt qu"`usr1` pour permettre un arrêt normal du service
— Mise à jour des dépendances externes
— Optimisation de l'utilisation de connexions à la base de données
— Les scripts linux `waarp-r66client` et `waarp-r66server` permettent de mettre à jour le modèle de données

8.13.5 Dépréciations

— L'option de configuration `dbcheck` est dépréciée

8.14 Waarp R66 3.0.12-1 (2019-05-10)

8.14.1 Correctifs

— Corrige des problèmes de perte de connexions à la base de données

8.15 Waarp R66 3.0.11-1 (2019-02-20)

8.15.1 Correctifs

— Correction du support des espaces dans les tâches TRANSFER
— Correction d'un NullPointerException au lancement du filewatcher
— Correctif dans le lancement des transferts asynchrones
— Suppression de la valeur miminum pour l'option runlimit
— Arrête l'envoi de paquets quand le transfert est stoppé ou annulé
— Correction de la commande exécutée sous Windows dans les tâches EXEC* si des slashes (« / ») sont utilisés dans le chemin de l'exécutable
— Ajout d'un délais de 5 minutes entre de tentatives de redémarrage du serveur R66 en cas d'échec de lancement dans les services systèmes (systemd et Windows).

8.15.2 Packaging

— `manager-send.sh` génère un fichier `get-files.list` pour Waarp Gateway SFTP : ce fichier est consommé par le script `waarp-get-sftp.sh` (livré avec les packages de la passerelle) pour interroger périodiquement les serveurs distants.
— `waarp-pull.sh` ne démarre plus qu'un seul transfert pour le fichiers disponibles.

8.16 Waarp R66 3.0.10-1 (2018-10-08)

8.16.1 Correctifs

— Support des espaces dans les tâches des chaînes de traitement
— Support des chemins UNC sous windows

8.17 Waarp R66 3.0.9-2 (2018-07-16)

8.17.1 Correctifs

— Correction de la gestion de la configuration des filewatchers par Manager
— Correction du redémarrage des filewatchers sous windows

8.18 Waarp R66 3.0.9 (2018-01-08)

8.18.1 Correctifs

— Mise à jour des dépendances externes
— Correction de l'erreur de chargement des données dans l'interface d'administration
— Le serveur Waarp R66 ne démarre plus si les ports sont déjà utilisés
— Les chemins de destination des tâches RENAME, MOVE, MOVERENAME, COPY, COPYRENAME peuvent contenir des espaces
— Correction du blocage des transferts asynchone quand leur nombre est supérieur à clientthread+11
— Correction d'un interblocage quand le nombre de transferts simultanés approche la valeur de clientthread
— Correction d'une fuite de mémoire
— Le Filewatcher ne démarrait pas quand fileinfo n'était pas renseigné dans le fichier de configuration

HTTP Routing Table

/log

/v2

Index

Symboles

A

C

D

H

I

J

M

O

P

R

S

T

V

W