Oracle® Endeca Information Discovery Integrator

Integrator ETL Server Guide

Version 3.2.0 • January 2016

ORACLE

Copyright and disclaimer

Oracle Endeca Supplement to Clover Server

This supplement provides specific information about support and limitations when using the Clover Server as the Oracle Endeca Integrator Server.

Supported Containers

Oracle Endeca Integrator Server is only supported in the following containers:

- Apache Tomcat
- Oracle WebLogic

While the Clover Server allows installation to other containers, installation into these containers is not supported for Oracle Endeca Integrator Server.

CloverETL Server: Reference Manual

This Reference Manual refers to CloverETL Server 4.1.x release.

Javlin

www.cloveretl.com

www.javlininc.com

Feedback welcome:

If you have any comments or suggestions for this documentation, please send them by email to `support@cloveretl.com`.

Table of Contents

Part I. CloverETL Server

Chapter 1. What is CloverETL Server?

The CloverETL Server is an enterprise runtime, monitoring, and automation platform for the CloverETL data integration suite. It provides the necessary tools to deploy, monitor, schedule, integrate, and automate data integration processes in large scale and complex projects.

CloverETL Server's HTTP and SOAP Web Services APIs provide additional automation control for integrating the CloverETL Server into existing application portfolios and processes.

The CloverETL Server is a Java application built to J2EE standards. We support a wide range of application servers including Apache Tomcat, Jetty, IBM WebSphere, Sun Glassfish, JBoss AS, and Oracle WebLogic.

Table 1.1. CloverETL Server and CloverETL Engine comparison

	CloverETL Server	CloverEngine as executable tool
possibilities of executing graphs	by calling http (or JMX, etc.) APIs (See details in Simple HTTP API (p. 156).)	by executing external process or by calling Java API
engine initialization	during server startup	init is called for each graph execution
thread and memory optimalization	threads recycling, graphs cache, etc.	not implemented
scheduling	scheduling by timetable, onetime trigger, logging included	external tools (i.e. Cron) can bes used
statistics	each graph execution has its own log file and result status is stored; each event triggered by the CS is logged	not implemented
monitoring	If graph fails, event listener will be notified. It may send email, execute shell command or execute another graph. See details in Graph Event Listeners (p. 137) Additionally server implements various APIs (HTTP and JMX) which may be used for monitoring of server/graphs status.	JMX mBean can be used while graph is running
storage of graphs and related files	graphs are stored on server file system in so called sandboxes	
security and authorization support	CS supports users/groups management, so each sandbox may have its own access privileges set. All interfaces require authentication. See details in Chapter 15, Server Side Job files - Sandboxes (p. 95).	passwords entered by user may be encrypted
integration capabilities	CS provides APIs which can be called using common protocols like HTTP. See details in Simple HTTP API (p. 156).	CloverEngine library can be used as embedded library in client's Java code or it may be executed as separated OS process for each graph.
development of graphs	CS supports team cooperation above one project (sandbox). CloverETL Designer is fully integrated with CloverETL Server (CS).	
scalability	CS implements horisontal scalability of transformation requests as well as data scalability. See details in Chapter 27, Clustering features (p. 183) In addition CloverEngine implements is vertical scalability natively.	Clover Engine implements vertival scalability
jobflow	CS implements various jobflow components. See details in the CloverETL manual.	Clover Engine itself has limited support of jobflow.

Part II. Installation Instructions

Chapter 2. System Requirements for CloverETL Server

Hardware Requirements

Table 2.1. Hardware requirements of CloverETL Server

	Standard Edition	Corporate Edition	Cluster
RAM	4 GB (recommended 16 GB)	8 GB (recommended 64 GB)	8 GB (recommended 64 GB)
Processors	up to 8 cores	16 cores	8 cores [a]
Disk space (installation)	1 GB	1 GB	1 GB
Disk space (tempspace)	> 25 GB [b]	> 25 GB [b]	> 25 GB [b]
Disk space (data)	> 50 GB [b]	> 50 GB [b]	> 50 GB [b]
Disk space (shared) [c]	-	-	> 50 GB [b]

[a] This may vary depending on total number of nodes and cores in license.
[b] Minimal value, the disk space depends on data.
[c] Disk space for shared sandboxes is required only for CloverETL Cluster.

Software Requirements

Operating system

- Microsoft Windows Server 2003/2008/2012 32/64 bit
- GNU/Linux 32/64 bit
- Mac OS X
- Unix
- HP-UX
- AIX
- IBM System i (also known as AS/400)

Java Virtual Machine

- Oracle JDK 7/8 32/64 bit
- IBM SDK 7 (for IBM WebSphere only)

Application Server

- Apache Tomcat 6 or 7 (p. 10)
- Jetty 9.1 (p. 14)
- IBM WebSphere 8.5 (p. 15)
- Glassfish 3.1 (p. 18)
- JBoss 6 or 7 (p. 19)
- Oracle WebLogic 11g (10.3.6) or 12c (12.1.2 or 12.1.3) 32/64 bit (p. 25)

Table 2.2. CloverETL Server Compatibility Matrix

	CloverETL 3.5	CloverETL 4.0	CloverETL 4.1	
Application Server	**Java 6 and 7**	**Java 7**	**Java 7**	**Java 8**
Tomcat 6	✔	✔	✔	✔
Tomcat 7	✖	✔	✔	✔
Tomcat 8	✖	✖	✔	✔
Jetty 6	✔	✖	✖	✖
Jetty 9	✖	✔	✔	✔
WebLogic 11g (10.3.6)	✔	✔	✔	✖
WebLogic 12c (12.1.2)	✔	✔	✔	✖
WebLogic 12c (12.1.3)	✖	✖	✔	✔
JBoss AS 5	✔	✖	✖	✖
JBoss AS 6	✔	✔	✔	✖
JBoss AS 7	✖	✔	✔[1]	✔[2]
Glassfish 2	✔	✖	✖	✖
Glassfish 3	✖	✔	✔	✖
WebSphere 7	✔	✖	✖	✖
WebSphere 8.5	✖	✔	✔	✖

[1] EAP 6.2
[2] EAP 6.4

We support Java 8 on particular supported application server only if the application server itself officially supports Java 8.

Configuration Repository

We support following database servers as a configuration repository. The officialy supported version, we are testing against, are in parentheses.

- MySQL (5.6.12) (p. 53)
- DB2 (10.5.1) (p. 54)
- Oracle (11.2.0.2.0) (p. 56)
- SQL Server 2008 (10.0.1600.22) (p. 57)
- PostgreSQL (9.2.4) (p. 58)

Firewall

- HTTP(S) incomming: Communication between Designer and Server
- JMX incomming: Tracking and debugging information
- Other outgoing: (depending on actual usage)
  - JDBC connection to databases (DBInputTable, DBOutputTable, DBExecute)
  - MQ (JMSReader, JMSWriter, JMS Listener)
  - HTTP(S) (Readers, WebserviceClient, HTTPConnector)
  - SMTP (EmailSender)
  - IMAP/POP3 (EmailReader)
  - FTP/SFTP/FTPS (readers, writers)

Chapter 3. Installing

The following sections describe two different installation types.

The section Evaluation Server (p. 7) details the quickest and simplest installation – without configuration.

The section Enterprise Server (p. 10) includes details about further testing and production on your chosen app-container and database.

To create a fully working instance of Enterprise CloverETL Server you should:

- install an application server
- create a database dedicated to CloverETL server
- set up limit on number of opened files and memory used
- install CloverETL Server into application server
- set up connection to database and mail server (optionaly encrypting passwords in configuration files)
- install license
- set up a master password for secure parameters
- configure temp space
- configure sandboxes

Evaluation Server

The default installation of **CloverETL Server** does not need any extra **database server**. It uses the embedded Apache Derby DB. What is more, it does not need any subsequent **configuration**. CloverETL Server configures itself during the first startup. Database tables and some necessary records are automatically created on first startup with an empty database. In the **Sandboxes** section of the web GUI, you can then check that sandboxes and a few demo graphs are created there.

Note

Default login credentials for CloverETL Server Console:

Username: **clover**

Password: **clover**

If you need to evaluate CloverETL Server features which need any configuration changes, e.g. sending emails, LDAP authentication, clustering, etc., or the CloverETL Server must be evaluated on another application container then Tomcat, please proceed with the common installation: Enterprise Server (p. 10)

Installation of Apache Tomcat

See Application Server (p. 5) system requirements for currently supported version of Apache Tomcat.

If you have Apache Tomcat already installed, you can skip this section.

1. Download the ZIP with binary distribution from http://tomcat.apache.org/download-60.cgi (Tomcat 6) or http://tomcat.apache.org/download-70.cgi (Tomcat 7).

   Tomcat may be installed as a service on Windows OS as well, however there may be some issues with access to file system, so it's not recommended aproach for evaluation. To install Apache Tomcat as **Windows Service** see Apache Tomcat as a Windows Service (p. 11).

2. After you download the zip file, unpack it.

3. Run Tomcat by `[tomcat_home]/bin/startup.sh` (or `[tomcat_home]/bin/startup.bat` on Windows OS).

4. Check whether Tomcat is running on URL: http://localhost:8080/. Apache Tomcat info page should appear.

5. Apache Tomcat is installed.

If in need of detailed installation instructions, go to: http://tomcat.apache.org/tomcat-6.0-doc/setup.html (Tomcat 6) or to http://tomcat.apache.org/tomcat-7.0-doc/setup.html (Tomcat 7)

Installation of CloverETL Server

1. Check if you meet prerequisites:

   - Oracle JDK or JRE v. 1.7.x or higher (check it by command **java -version**)
   - `JAVA_HOME` or `JRE_HOME` environment variable has to be set.

     Unix-like systems:

     Check it by **echo $JAVA_HOME** and **echo $JAVA_HOME** commands.

     It can be set up in the file `[tomcat]/bin/setenv.sh` from step 2 by inserting line to the beginning of the file, e.g.

```
export JAVA_HOME=/opt/jdk1.7.0_60
```

     Windows system:

     Check it by **echo %JAVA_HOME%** and **echo %JRE_HOME%** commands.

     It can be set up in the file `[tomcat]/bin/setenv.bat` from step 2 by inserting line to the beginning of the file, e.g.

```
set JAVA_HOME=C:\Program Files\Java\jdk1.7.0_60
```

   - Apache Tomcat 6.0.x or 7.0.x is installed. See Installation of Apache Tomcat (p. 7) for details.

2. Set memory limits and other switches. See section Memory Settings (p. 37) for details.

   Create setenv file:

   Unix-like systems: `[tomcat]/bin/setenv.sh`

```
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=512m -Xms128m -Xmx2048m"
export CATALINA_OPTS="$CATALINA_OPTS -Dderby.system.home=$CATALINA_HOME/temp -server"
echo "Using CATALINA_OPTS: $CATALINA_OPTS"
```

   Windows systems: `[tomcat]/bin/setenv.bat`

```
set CATALINA_OPTS=%CATALINA_OPTS% -XX:MaxPermSize=512m -Xms128m -Xmx1024m
set CATALINA_OPTS=%CATALINA_OPTS% -Dderby.system.home=%CATALINA_HOME%/temp -server
echo "Using CATALINA_OPTS: %CATALINA_OPTS%"
```

3. Download the web archive file (`clover.war`) containing CloverETL Server for Apache Tomcat and `clover-license.war` containing valid license.

4. Deploy both WAR files: `clover.war` and `clover-license.war` to `[tomcat_home]/webapps` directory.

To avoid deployment problems, Tomcat should be down during the copying.

5. Run Tomcat by `[tomcat_home]/bin/startup.sh` (or `[tomcat_home]/bin/startup.bat` on Windows OS).

6. Check whether CloverETL Server is running on URLs:

   - Web-app root

     `http://[host]:[port]/[contextPath]`

     The default Tomcat port for the http connector is 8080 and the default `contextPath` for CloverETL Server is "clover", thus the default URL is:

     http://localhost:8080/clover/

   - Web GUI

     `http://[host]:[port]/[contextPath]/gui` http://localhost:8080/clover/gui

     Use default administrator credentials to access the web GUI: username - "clover", password - "clover".

7. CloverETL Server is now installed and prepared for basic evaluation. There are couple of sandboxes with various demo transformations installed.

Enterprise Server

This section describes instalation of CloverETL Server on various app-containers in detail, also describes the ways how to configure the server. If you need just quickly evaluate CloverETL Server features which don't need any configuration, evaluation installation may be suitable: Evaluation Server (p. 7)

CloverETL Server for enterprise environment is shipped as a *Web application archive* (WAR file). Thus standard methods for deploying a web application on you application server may be used. However each application server has specific behavior and features. Detailed information about their installation and configuration can be found in the chapters below.

List of suitable containers

In case of problems during your installation see Possible issues during installation (p. 32).

Apache Tomcat

Installation of Apache Tomcat

See Application Server (p. 5) in system requirements for currently supported version of Apache Tomcat.

If you have Apache Tomcat already installed, you can skip this section.

In case of installation as **Windows Service** skip this section and continue with Apache Tomcat as a Windows Service (p. 11).

1. Download the binary distribution from http://tomcat.apache.org/download-60.cgi (Tomcat 6) or http://tomcat.apache.org/download-70.cgi (Tomcat 7).

2. After you download the zip file, unpack it.

3. Run Tomcat by `[tomcat_home]/bin/startup.sh` (or `[tomcat_home]/bin/startup.bat` on Windows OS).

4. Check whether Tomcat is running on URL: http://localhost:8080/. Apache Tomcat info page should appear.

5. Apache Tomcat is installed.

If in need of detailed installation instructions, go to: http://tomcat.apache.org/tomcat-6.0-doc/setup.html (Tomcat 6) or to http://tomcat.apache.org/tomcat-7.0-doc/setup.html (Tomcat 7)

Continue with: Installation of CloverETL Server (p. 12)

In case of installation on **IBM AS/400** continue with Apache Tomcat on IBM AS/400 (iSeries) (p. 12).

Apache Tomcat as a Windows Service

1. Download the binary distribution from http://tomcat.apache.org/download-60.cgi (Tomcat 6) or http://tomcat.apache.org/download-70.cgi (Tomcat 7). Use the file called **Windows Service Installer**.

2. Use the standard installation wizard to install Apache Tomcat.

Important

The instructions mentioned bellow assume that Tomcat (as a service) is already installed. If this is not true, it is needed to do this first (using of Tomcat's *Windows Service Installer* is recommended).

Set up `JAVA_HOME` to point to the correct Java version: rigth click on **This computer** (or **Computer** on some Windows versions) and choose **Properties** →**Advanced System Settings** →**Advanced** →**Environment variables**. The restart of operating system is needed to apply changes.

In case that Tomcat is installed as a Windows service, CloverETL configuration is performed using configuration of the respective service. The configuration can be performed either by graphical utility `[tomcat_home]/bin/Tomcat7w.exe` or by command line utility `[tomcat_home]/bin/Tomcat7.exe`.

Graphical configuration utility

When using *graphical* configuration utility, select **Java** tab and set initial and maximum heap size in **Initial memory pool** and **Maximum memory pool** fields. Other configuration parameters can be placed in **Java Options** field, being separated by new line. Then click on **Apply** and restart the service.

The **Java** tab allows you to use alternative Java virtual machine by setup of path to `jvm.dll` file.

Command line tool

When using *command line* tool, navigate to `[tomcat_home]/bin` and stop the service (if running) using (supposing that service is named "Tomcat7"):

```
.\Tomcat7.exe //SS//Tomcat7
```

Then configure service using command:

```
.\Tomcat7.exe //US//Tomcat7 --JvmMs=512 --JvmMx=1024
  --JvmOptions=-Dclover.config.file=C:\path\to\clover-config.properties#-XX:MaxPermSize=256m
```

The parameter JvmMs is the initial and JvmMx is the maximum heap size in MB, JVM options are separated by '#' or ';'. Finally start the service from Windows administration console or using:

```
.\Tomcat7.exe //TS//Tomcat7
```

When Apache Tomcat is run as a Windows service, it is not automatically available for tools like JConsole or JVisualVM to attach and monitor Java process. However those tools can still connect to the process via JMX. In order to expose Tomcat's Java process via JMX add the following options to the service settings:

```
-Dcom.sun.management.jmxremote.port=3333
        -Dcom.sun.management.jmxremote.ssl=false
        -Dcom.sun.management.jmxremote.authenticate=false
```

Once the service is run with those options you can connect to port 3333 using JMX and monitor server.

Note

The same instructions can be used with Tomcat 6, just replace the number 7 with the number 6.

More information about run of Java applications as Windows Service can be found at http://commons.apache.org/proper/commons-daemon/procrun.html

Continue with: Installation of CloverETL Server (p. 12)

Apache Tomcat on IBM AS/400 (iSeries)

To run CloverETL Server on the iSeries platform, there are some additional settings:

1. Declare you are using Java 6.0 32-bit
2. Run Java with parameter `-Djava.awt.headless=true`

To configure this you can modify/create a file `[tomcat_home]/bin/setenv.sh` which contains:

```
JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/jdk70/32bit
```

```
JAVA_OPTS="$JAVA_OPTS -Djava.awt.headless=true"
```

Continue with: Installation of CloverETL Server (p. 12)

Installation of CloverETL Server

1. Download the web archive file (`clover.war`) containing CloverETL Server for Apache Tomcat.
2. Check if you meet prerequisites:
   - Oracle JDK or JRE v. 1.7.x or higher
   - `JAVA_HOME` or `JRE_HOME` environment variable has to be set.
   - Apache Tomcat 6.0.x or 7.0.x is installed. CloverETL Server is developed and tested with the Apache Tomcat 6.0.x and 7.0.x containers (it may work unpredictably with other versions). See Installation of Apache Tomcat (p. 10) for details.
   - It is strongly recommended you change default limits for the `heap` and `perm gen` memory spaces.

     See section Memory Settings (p. 37) for details.

     You can set the minimum and maximum memory heap size by adjusting the "Xms" and "Xmx" JVM parameters. You can set JVM parameters for Tomcat by setting the environment variable `JAVA_OPTS` in the `[TOMCAT_HOME]/bin/setenv.sh` file (if it does not exist, you may create it).

     Create setenv file:

     Unix-like systems: `[tomcat]/bin/setenv.sh`

```
export CATALINA_OPTS="$CATALINA_OPTS -XX:MaxPermSize=512m -Xms128m -Xmx1024m"
export CATALINA_OPTS="$CATALINA_OPTS -Dderby.system.home=$CATALINA_HOME/temp -server"
echo "Using CATALINA_OPTS: $CATALINA_OPTS"
```

     Windows systems: `[tomcat]/bin/setenv.bat`

```
set "CATALINA_OPTS=%CATALINA_OPTS% -XX:MaxPermSize=512m -Xms128m -Xmx1024m"
set "CATALINA_OPTS=%CATALINA_OPTS% -Dderby.system.home=%CATALINA_HOME%/temp -server"
echo "Using CATALINA_OPTS: %CATALINA_OPTS%"
```

As visible in the settings above, there is also switch `-server`. For performance reasons, it is recommended to run the container in the "server" mode.

3. Copy `clover.war` (which is built for Tomcat) to `[tomcat_home]/webapps` directory.

   Please note, that copying is not an atomic operation. If Tomcat is running, mind duration of the copying process! Too long copying might cause failure during deployment as Tomcat tries to deploy an incomplete file. Instead, manipulate the file when the Tomcat is not running.

4. War file should be detected and deployed automatically without restarting Tomcat.

5. Check whether CloverETL Server is running on URLs:

   - Web-app root

     `http://[host]:[port]/[contextPath]`

     The default Tomcat port for the http connector is 8080 and the default `contextPath` for CloverETL Server is "clover", thus the default URL is:

     http://localhost:8080/clover/

   - Web GUI

     `http://[host]:[port]/[contextPath]/gui`

     The default Tomcat port for the http connector is 8080 and the default contextPath for CloverETL Server is "clover", thus the default URL is:

     http://localhost:8080/clover/gui

     Use default administrator credentials to access the web GUI: user name - "clover", password - "clover".

Continue with: Configuration of CloverETL Server on Apache Tomcat (p. 13)

Configuration of CloverETL Server on Apache Tomcat

Default installation (without any configuration) is recommended only for evaluation purposes. For production, at least DB data source and SMTP server configuration is recommended.

The connection to database is configured in **property file**. On Apache Tomcat, it is `$CATALINA_HOME/webapps/clover/WEB-INF/config.properties` file.

Properties File on Specified Location

Example of such a file:

```
jdbc.driverClassName=...
jdbc.url=...
jdbc.username=...
jdbc.password=...
jdbc.dialect=...
license.file=/path/to/license.dat
```

Properties file is loaded from location specified by the system property or environment variable `clover_config_file` (`clover.config.file`).

On Apache Tomcat, you can set the system property in the `[TOMCAT_HOME]/bin/setenv.sh` file (if it does not exist, you may create it). Just add: `JAVA_OPTS="$JAVA_OPTS -Dclover_config_file=/path/to/cloverServer.properties"`.

If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 51).

Continue with: Installation of CloverETL Server License (p. 27)

Jetty

Installation of CloverETL Server (p. 14)
Configuration of CloverETL Server on Jetty (p. 14)

Installation of CloverETL Server

1. Download the web archive file (`clover.war`) containing the CloverETL Server application which is built for Jetty.
2. Check if prerequisites are met:
   - Oracle JDK or JRE (See Java Virtual Machine (p. 5) for required Java version.)
   - See Application Server (p. 5) in system requirements for curently supported versions of **Jetty**.

     All Jetty releases are available from http://download.eclipse.org/jetty/.
   - Memory allocation settings

     It involves JVM parameters: -Xms -Xmx (heap memory) and -XX:MaxPermSize (classloaders memory limit). See section Memory Settings (p. 37) for details.

     You can set the parameters by adding

```
JAVA_OPTIONS='$JAVA_OPTIONS -Xms128m -Xmx1024m -XX:MaxPermSize=256m'
```

     to `[jetty-home]/bin/jetty.sh`

     On Windows OS, edit `[jetty-home]/start.ini` and add those parameters on the end of the file, each parameter on a new line.
3. Copy `clover.war` to `[jetty-home]/webapps`.
4. Run `[jetty-home]/bin/jetty.sh start` (or `java -jar start.jar --exec` on Windows OS).

Finally, you can check if the server is running e.g. on http://localhost:8080/clover/.

Configuration of CloverETL Server on Jetty

Default installation (without any configuration) is recommended only for evaluation purposes. For production, at least DB data source and SMTP server configuration is recommended.

There are more ways how to set config properties, yet the most common one is properties file in a specified location.

Properties file in Specified Location

Example of such a file:

```
jdbc.driverClassName=...
jdbc.url=...
jdbc.username=...
jdbc.password=...
jdbc.dialect=...
license.file=/path/to/license.dat
```

Note

JDBC Driver must be JDBC 4 compliant and stored in the `[jetty-home]/lib/ext`.

The common properties file is loaded from a location which is specified by the environment/system property `clover_config_file` or `clover.config.file`. This is a recommended way of configuring Jetty.

On Unix-like OS, you can set system property in the `[jetty-home]/bin/jetty.sh` file. Add:

```
JAVA_OPTIONS="$JAVA_OPTIONS -Dclover_config_file=/path/to/cloverServer.properties"
```

On Windows OS add the property to `[jetty-home]/start.ini`, just after the memory settings.

Continue with: Installation of CloverETL Server License (p. 27)

IBM WebSphere

Installation of CloverETL Server (p. 15)
Configuration of CloverETL Server on IBM WebSphere (p. 17)

Installation of CloverETL Server

1. Get the web archive file (`clover.war`) with CloverETL Server application which is built for WebSphere.
2. Check if you meet prerequisites:
   - IBM Java SDK (See Java Virtual Machine (p. 5) for required Java version.)

     Important

     In order to ensure reliable function of CloverETL Server always use the latest version of IBM Java SDK. At least SDK 7.0 SR6 (package *IBM WebSphere SDK Java Technology Edition V7.0.6.1*) is recommended. Using older SKDs may lead to deadlocks during execution of specific ETL graphs.

   - See Application Server (p. 5) in system requirements for currently supported version of IBM WebSphere. (see http://www.ibm.com/developerworks/downloads/ws/was/)
   - Memory allocation settings

     It involves JVM parameters: -Xms and -Xmx See section Memory Settings (p. 37) for details.

     You can set heap size in IBM WebSphere's **Integrated Solutions Console** (by default accessible at: `http://[host]:9060/ibm/console/`)

- Go to **Servers →Server Types →WebSphere application servers →[server1] (or another name of your server) →Java and Process Management →Process definition →Java Virtual Machine**
- There is the **Maximum heap size** field. Default value is only 256 MB, which is not enough for ETL transformations.

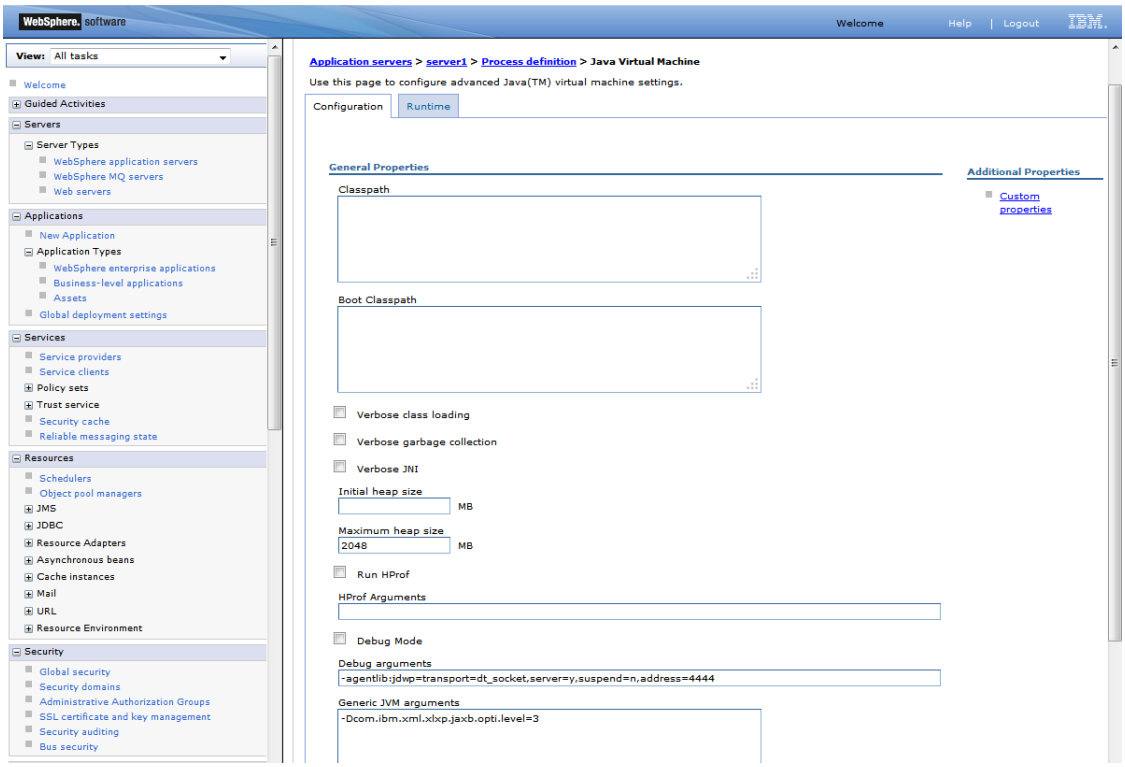


Figure 3.1. Adjusting Maximum heap size limit

- On the same page, there is **Generic JVM arguments**. Add the perm space limit there, e.g. like this:

  ```
  -XX:MaxPermSize=512M
  ```

  Add the direct memory limit, e.g. like this:

  ```
  -XX:MaxDirectMemorySize=512M
  ```

- Java runtime settings

  Go to **Servers →Application servers →[server1] (or another name of your server) →Java SDKs** Here select version 1.7 as the default SDK.

- Save the changes to configuration and restart the server so that they take effect.

3. Deploy WAR file

   - Go to **Integrated Solutions Console**

     (http://localhost:9060/ibm/console/)

   - Go to **Applications →New Application →New Enterprise Application** Here select a WAR archive of the CloverETL server and deploy it to the application server, but do not start it.

4. Configure application class loading

Go to **WebSphere Enterprise Applications →clover_war (or other name of the Clover application) →Manage Modules →CloverETL** Here under **Class loader order** select **Classes loaded with local class loader first (parent last)**.

5. Save the changes to the server configuration and start the **clover_war** application.
6. Check whether the server is running

   Provided you set `clover.war` as the application running with "clover" context path. Notice the port number has changed:

   http://localhost:9080/clover

Note

Please note, that some CloverETL features using third party libraries don't work properly on IBM WebSphere

- Hadoop is guaranteed to run only on Oracle Java 1.6+, but Hadoop developers do make an effort to remove any Oracle/Sun-specific code. See Hadoop Java Versions on Hadoop Wiki.
- OSGi framework (if configured and initialized) causes malfunction of WebServiceClient and EmailReader components. See Embedded OSGi Framework (p. 177) for details.
- **AddresDoctor5** on IBM WebSphere, requires additional JVM parameter `-Xmso2048k` to prevent AddressDoctor from crashing JVM. See documentation of AddressDoctor component.

Configuration of CloverETL Server on IBM WebSphere

Default installation (without any configuration) is recommended only for evaluation purposes. For production, configuring at least the DB data source and SMTP server is recommended.

There are more ways how to set config properties. The most common one is properties file in a specified location.

Properties File in Specified Location

Example of such a file:

```
jdbc.driverClassName=...
jdbc.url=...
jdbc.username=...
jdbc.password=...
jdbc.dialect=...
license.file=/path/to/license.dat
```

Set system property (or environment variable) `clover_config_file` pointing to the config properties file.

- go to **Integrated Solutions Console**

  (http://localhost:9060/ibm/console/)

- Go to **Servers →WebSphere application servers →[server-name] →Java and Process Management →Process Definition →Java Virtual Machine →Custom Properties**
- Create system property named `clover_config_file` whose value is full path to config file named e.g. `cloverServer.properties` on your file system.
- This change requires restarting IBM WebSphere.

Continue with: Installation of CloverETL Server License (p. 27)

Glassfish / Sun Java System Application Server

Installation of CloverETL Server (p. 18)
Configuration of CloverETL Server on Glassfish (p. 18)

Important

Glassfish 3.1.2 contains a bug causing **Launch Services** to work improperly (see https://java.net/jira/browse/GLASSFISH-18444). Use a different version of Glassfish (version 3.1.2.2 is recommended).

Installation of CloverETL Server

1. Get CloverETL Server web archive file (`clover.war`) which is built for Glassfish 3.
2. Check if you meet prerequisites
   - Oracle JDK or JRE (See Java Virtual Machine (p. 5) for required java version.)
   - Glassfish (CloverETL Server is tested with version 3.1.2.2)
   - Memory allocation settings

     It involves JVM parameters: `-Xms` `-Xmx` and `-XX:MaxPermSize` See section Memory Settings (p. 37) for details.

     You can set the heap size and perm space in XML file `[glassfish-home]/glassfish/domains/domain1/config/domain.xml` Add or change these sub-elements to `<java-config>`:

```
<jvm-options>-XX:MaxPermSize=384m</jvm-options>
<jvm-options>-XX:PermSize=256m</jvm-options>
<jvm-options>-Xms512m</jvm-options>
<jvm-options>-Xmx2g</jvm-options>
```

     These changes require restarting Glassfish to take effect.
3. Deploy WAR file
   - Open **Glassfish Administration Console**

     It is accessible at http://localhost:4848/ by default.
   - Go to **Applications** →**Web Applications** and click **Deploy ...**.
   - Upload WAR file woth CloverETL server application or select the file from filesystem if it present on the machine running Glassfish.
   - Fill in attributes **Application name** and **Context Root** with "clover".
   - Submit form.

Configuration of CloverETL Server on Glassfish

Default installation (without any configuration) is recommended only for evaluation purposes. For production, configuring at least the DB data source and SMTP server is recommended.

There are more ways how to set config properties. The most common one is properties file in a specified location.

Properties File in Specified Location

Example of such a file:

```
datasource.type=JDBC
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost:5432/cloverdb
jdbc.username=root
jdbc.password=
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
license.file=/path/to/license.dat
```

Set system property `clover.config.file` pointing to the config properties file:

- Go to **Glassfish Administration Console**

  By default accessible at http://localhost:4848/

- Go to **Configuration →System Properties**
- Create system property named `clover.config.file` whose value is full path to a file on your file system named e.g. `/home/clover/cloverServer.properties`.

Copy the JDBC driver JAR file for selected database (PostgreSQL) into `[glassfish]/glassfish/domains/[domain-name]/lib`

These changes require restarting Glassfish to take effect.

Continue with: Installation of CloverETL Server License (p. 27)

JBoss Application Server

Installation of CloverETL Server (p. 19)
Configuration of CloverETL Server on JBoss AS (p. 20)

Installation of CloverETL Server

1. Get CloverETL Server web archive file (`clover.war`) that is built for JBoss AS.
2. Check if you meet prerequisites
   - Oracle JDK or JRE (See Java Virtual Machine (p. 5) for required Java version.)
   - JBoss AS 6.x http://www.jboss.org/jbossas/downloads
   - Memory settings for JBoss Java process. See section Memory Settings (p. 37) for details.

     You can set the memory limits in `[jboss-home]/bin/run.conf` (`run.conf.bat` on Windows OS)

```
JAVA_OPTS="$JAVA_OPTS -XX:MaxPermSize=512m -Xms128m -Xmx2048m"
```

     On Windows, perform steps analogic to the ones above.
3. Create separated JBoss server configuration

However it may be useful to use specific JBoss server configuration, when it is necessary to run CloverETL:

- isolated from other JBoss applications
- with different set of services
- with different libraries on the classpath than other applications

See the JBoss manual for details about JBoss server configuration: JBoss Server Configurations Start the Server With Alternate Configuration

4. Configure database connection

   As CloverETL Server's embedded Derby database does not work under JBoss AS, a database connection has to be always configured. We used MySQL accessed via JNDI-bound datasource in this example:

   - Create datasource deployment file `[jboss-home]/server/[serverConfiguration]/deploy/mysql-ds.xml`

```
<datasources>
    <local-tx-datasource>
        <jndi-name>CloverETLServerDS</jndi-name>
        <connection-url>jdbc:mysql://localhost:3306/cloverServerDB</connection-url>
        <driver-class>com.mysql.jdbc.Driver</driver-class>
        <user-name>root</user-name>
        <password>root</password>
    </local-tx-datasource>
</datasources>
```

Note

Special characters in the XML file have to be typed in as XML entities. For instance, ampsersand "&" as "&" etc.

   "CloverETLServerDS" is the name under which the datasource will be accessible. The thing to do here is to set database connection parameters (`connection-url` , `driver-class` , `user-name` and `password`) to the database. The database has to be empty before the first execution, the server creates its tables on its own.

   - Put the JDBC driver JAR file for your database to the application server classpath. In this example we copied the file `mysql-connector-java-5.1.5-bin.jar` to `[jboss-home]/server/[serverConfiguration]/lib`

5. Configure CloverETL Server according to description in the next section (p. 20) .

6. Deploy WAR file

   Copy `clover.war` to `[jboss-home]/server/[serverConfiguration]/deploy`

7. Start JBoss AS via `[jboss-home]/bin/run.sh` (or `run.bat` on Windows OS) If you want to run JBoss with specific server configuration, it has to be specified as parameter like this: `[jboss-home]/bin/run.sh -c [serverConfiguration]` If the serverConfiguration isn't specified, the "default" is used.

Configuration of CloverETL Server on JBoss AS

Default installation (without any configuration) does not work under JBoss AS. In order to be able to use the CloverETL Server, working database connection is required.

There are more ways how to set configuration properties. The most common one is a properties file in a specified location.

Properties File in Specified Location

- Create `cloverServer.properties` in a suitable (i.e. readable by JBoss AS process) directory

```
datasource.type=JNDI
datasource.jndiName=java:/CloverETLServerDS
jdbc.dialect=org.hibernate.dialect.MySQLDialect
license.file=/home/clover/config/license.dat
```

  `datasource.type` indicates the server will use JNDI-bound datasource created in steps above. The property `datasource.jndiName` specifies where is the datasource to be found in JNDI. Set JDBC dialect according to your database server (Part III, "Configuration" (p. 40)). You can set path to your license file, too.

- Set system property (or environment variable) `clover.config.file` .

  It should contain the full path to the `cloverServer.properties` file created in the previous step.

  The simplest way is setting Java parameter in `[jboss-home]/bin/run.sh`, e.g.:

```
export JAVA_OPTS="$JAVA_OPTS -Dclover.config.file=/home/clover/config/cloverServer.properties"
```

  Do not override other settings in the `JAVA_OPTS` property - i.e. memory settings described above.

  On Windows OS, edit `[jboss-home]/bin/run.conf.bat` and add this line to the section where options are passed to the JVM:

```
set JAVA_OPTS=%JAVA_OPTS% -Dclover.config.file=C:/JBoss6/cloverServer.properties
```

- Restart the JBoss AS so that the changes take effect.
- Check the CloverETL Server application is running:

  Server's console is accessible at http://localhost:8080/clover by default.

Continue with: Installation of CloverETL Server License (p. 27)

JBoss Enterprise Application Platform

Installation of CloverETL Server (p. 21)
Configuration of CloverETL Server on JBoss EAP (p. 23)

Installation of CloverETL Server

1. Get CloverETL Server web archive file (`clover.war`) which is built for JBoss EAP.
2. Check if you meet prerequisites
   - Oracle JDK or JRE (See Java Virtual Machine (p. 5) for required Java version.)

- JBoss EAP 6.x (JBoss AS 7) - see http://www.jboss.org/jbossas/downloads
- Memory settings for JBoss Java process. See section Memory Settings (p. 37) for details.

  You can set the memory limits in `[jboss-home]/bin/standalone.conf` (`[jboss-home]/bin/standalone.conf.bar` on Windows OS) for JBoss EAP in standalone mode:

```
JAVA_OPTS="$JAVA_OPTS -XX:MaxPermSize=512m -Xms128m -Xmx2048m"
```

  On Windows, perform steps analogic to the ones above.

3. Configure database connection

   By default CloverETL Server uses embedded Derby database, however such setup is not recommended for production use. You can use database connection provided by JNDI-bound datasource deployed by JBoss EAP. In order to define the datasource, edit the file `[jboss-home]/standalone/configuration/standalone.xml` and into section `<subsystem xmlns="urn:jboss:domain:datasources:1.1">` under element `<datasources>` add definition of the datasource. Here is an example of datasource connecting to MySQL database:

```
<datasource jndi-name="java:jboss/datasources/CloverETLServerDS"
    pool-name="CloverETLServerDS-Pool" enabled="true">
    <connection-url>jdbc:mysql://localhost:3307/cloverServerDB</connection-url>
    <driver>com.mysql</driver>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
    <pool>
        <min-pool-size>5</min-pool-size>
        <max-pool-size>50</max-pool-size>
        <prefill>true</prefill>
    </pool>
    <security>
        <user-name>root</user-name>
        <password>root</password>
    </security>
    <statement>
        <prepared-statement-cache-size>32</prepared-statement-cache-size>
        <share-prepared-statements>true</share-prepared-statements>
    </statement>
</datasource>
<drivers>
    <driver name="com.mysql" module="mysql.driver">
        <driver-class>com.mysql.jdbc.Driver</driver-class>
    </driver>
</drivers>
```

4. The datasource definition references a module (`mysql.driver`) with MySQL JDBC driver. Take following steps to add the module:

5. Under JBoss EAP there are more options to setup CloverETL Server's database: along with JNDI-bound data source it is possible to use embedded Derby database or other supported database specified in CloverETL configuration file.

   In order to be able to connect to the database, one needs to define global module so that the driver is available for CloverETL web application - copying the driver to the `lib/ext` directory of the server will not work. Such module is created and deployed in few steps (the example is for MySQL and module's name is `mysql.driver`):

   - Create directory `[jboss-home]/modules/mysql/driver/main` (note that the directory path corresponds to module name `mysql.driver`)

- Copy the driver `mysql-connector-java-5.1.5-bin.jar` to that directory and create there file `module.xml` with following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.1" name="mysql.driver">
    <resources>
        <resource-root path="mysql-connector-java-5.1.5-bin.jar" />
    </resources>
    <dependencies>
        <module name="javax.api" />
    </dependencies>
</module>
```

- Add the module to global server modules: in case of the standalone JBoss EAP server they are defined in `[jboss-home]/standalone/configuration/standalone.xml` . The module is to be added into EE domain subsystem section:

```
<subsystem xmlns="urn:jboss:domain:ee:1.1">
    <global-modules>
        <module name="mysql.driver" slot="main" />
    </global-modules>
    <spec-descriptor-property-replacement>false</spec-descriptor-property-replacement>
    <jboss-descriptor-property-replacement>true</jboss-descriptor-property-replacement>
</subsystem>
```

6. Configure CloverETL Server according to description in the next section (p. 23) .

7. Deploy WAR file

   Copy the file `clover.war` to `[jboss-home]/standalone/deployments`

8. Run `[jboss-home]/bin/standalone.sh` (or `standalone.bat` on Windows OS) to start the JBoss platform.

   It may take a couple of minutes until all applications are started.

9. Check JBoss response and CloverETL Server response

   - JBoss administration console is accessible at http://localhost:8080/ by default. Default username/password is admin/admin

   - CloverETL Server is accessible at http://localhost:8080/clover by default.

Configuration of CloverETL Server on JBoss EAP

Default installation (without any configuration) is recommended only for evaluation purposes. For production use, configuring at least the database connection and SMTP server connection is recommended.

There are more ways how to set config properties. The most common one is properties file in a specified location.

Properties File in Specified Location

- Create `cloverServer.properties` in a directory the JBoss EAP has right to read:

```
datasource.type=JNDI
datasource.jndiName=java:jboss/datasources/CloverETLServerDS
jdbc.dialect=org.hibernate.dialect.MySQLDialect
license.file=/home/clover/config/license.dat
```

Do not forget to set correct JDBC dialect according to your database server (Part III, "Configuration"(p. 40)). You can set the path to the license file, too.

- Alternatively, you can set "JDBC" `datasource.type` and configure the database connection to be managed directly by CloverETL Server (provided that you have deployed proper JDBC driver module to the server):

```
datasource.type=JDBC
jdbc.url=jdbc:mysql://localhost:3306/cloverServerDB
jdbc.dialect=org.hibernate.dialect.MySQLDialect
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.username=root
jdbc.password=root
license.file=/home/clover/config/license.dat
```

- Set system property (or environment variable) `clover.config.file` .

  It should contain the full path to the `cloverServer.properties` file created in the previous step.

  The simplest way to set the system property is to edit the configuration file `[jboss-homme]/standalone/configuration/standalone.xml` , just under section `<extensions>` add following snippet:

```
<system-properties>
    <property name="clover.config.file" value="C:/jboss-eap-6.2/cloverServer.properties" />
</system-properties>
```

- Restart the JBoss EAP so that the changes take effect.
- Check the CloverETL Server application is running:

  Server's console is accessible at http://localhost:8080/clover by default.

Note

The JBoss EAP has, by default, enabled HTTP session replication. This requires session serialization that is not supported by CloverETL Server, and produces lots of harmless errors in JBoss's console like this:

```
10:56:38,248 ERROR [org.infinispan.transaction.TransactionCoordinator] (http-/127.0.0.1:8080-2)
ISPN000188: Error while processing a commit in a two-phase transaction:
java.lang.UnsupportedOperationException: Serialization of HTTP session objects is not supported
by CloverETL Server - disable the session passivation/replication for this web application.
        at com.cloveretl.server.web.gui.e.writeExternal(Unknown Source) [cs.jar:]
        at org.jboss.marshalling.river.RiverMarshaller.doWriteObject(RiverMarshaller.java:874)
```

To get rid of these errors, edit `[jboss-home]/standalone/configuration/standalone.xml` and under section `<subsystem`

`xmlns="urn:jboss:domain:infinispan:1.5">` comment out whole block `<cache-container name="web" aliases="standard-session-cache">` disabling the session replication.

Continue with: Installation of CloverETL Server License (p. 27)

Oracle WebLogic Server

Installation of CloverETL Server (p. 25)
Configuration of CloverETL Server on WebLogic (p. 26)

Installation of CloverETL Server

1. Get CloverETL Server web archive file (`clover.war`) which is built for WebLogic.
2. Check if you meet prerequisites
   - Oracle JDK or JRE (See Java Virtual Machine (p. 5) for required Java version.)
   - WebLogic (CloverETL Server is tested with WebLogic Server 11g (10.3.6) and WebLogic Server 12c (12.1.2), see http://www.oracle.com/technetwork/middleware/ias/downloads/wls-main-097127.html)

     WebLogic has to be running and a domain has to be configured. You can check it by connecting to **Administration Console**: http://hostname:7001/console/ (7001 is the default port for HTTP). Username and password are specified during installation.
   - Memory allocation settings

     It involves JVM parameters: -Xms -Xmx and -XX:MaxPermSize

     See section Memory Settings (p. 37) for details.

     You can set it i.e. by adding

```
export JAVA_OPTIONS='$JAVA_OPTIONS -Xms512m -Xmx2048m -XX:MaxPermSize=512m' to the start script
```

     This change requires restarting the domain.
3. Change HTTP Basic Authentication configuration
   - When WebLogic finds "Authentication" header in an HTTP request, it tries to find a user in its own realm. This behavior has to be disabled so CloverETL could auhenticate users itself.
   - Modify config file `[domainHome]/config/config.xml`. Add element: `<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>` into element `<security-configuration>` (just before the end tag).
4. Deploy WAR file (or application directory)
   - Get a clover.war suitable for your WebLogic version.
   - Deploy the clover.war using the **WebLogic Server Administration Console**. See the Oracle Fusion Middleware Administrator's Guide (http://docs.oracle.com/cd/E23943_01/core.1111/e10105/toc.htm) for details.
5. Configure license and other configuration properties. (See Configuration of CloverETL Server on WebLogic (p. 26))

6. Check CloverETL Server URL

   - Web-app is started automatically after deploy, so you can check whether it is up and running.

     CloverETL Server is accessible at http://host:7001/clover by default. Port 7001 is the default WebLogic HTTP Connector port.

Configuration of CloverETL Server on WebLogic

Default installation (without any configuration) is recommended only for evaluation purposes. For production, at least the DB data source and SMTP server configuration is recommended.

There are more ways how to set config properties. The most common one is properties file in a specified location.

Properties File in Specified Location

Create `cloverServer.properties` in a suitable directory.

Config file should contain DB datasource config, SMTP connection config, etc.

Set system property (or environment variable) `clover_config_file` pointing to the config properties file

- Set JAVA_OPTIONS variable in the WebLogic domain start script `[domainHome]/startWebLogic.sh`

```
JAVA_OPTIONS="${JAVA_OPTIONS} -Dclover_config_file=/path/to/clover-config.properties
```

- This change requires restarting WebLogic.

Continue with: Installation of CloverETL Server License (p. 27)

Installation of CloverETL Server License

To be able to execute graphs, CloverETL Server requires a valid license. You can install CloverETL Server without any license, but no graph will be executed.

There are three ways of installing license working on all application servers.

- The simplest way is using 'Add license form' and save license content to database. See Installation of CloverETL Server License using a Web Form (p. 27).
- The second way is configuring `license.file` property to set the plain license file. See Installation of CloverETL Server License using a license.file property (p. 30).
- The last way is a separate web application `clover-license.war`. See Separate License WAR (p. 30).

Note

CloverETL license can be **changed** any time by replacing `license.dat` file. Afterwards, you have to let CloverETL Server know the license has changed.

- Go to **server web GUI →Configuration →Setup →License**
- Click **Reload license**.
- Alternatively, you can restart the CloverETL Server application.

Note

All three ways listed below can be used at the same time. The most recent valid license is used.

Installation of CloverETL Server License using a Web Form

If the CloverETL Server has been started without assigning any license, you can use **Add license** form in the server gui to install it. In this case the hyperlink *No license available in system. Add new license* is displayed on the login page. It links to the form.

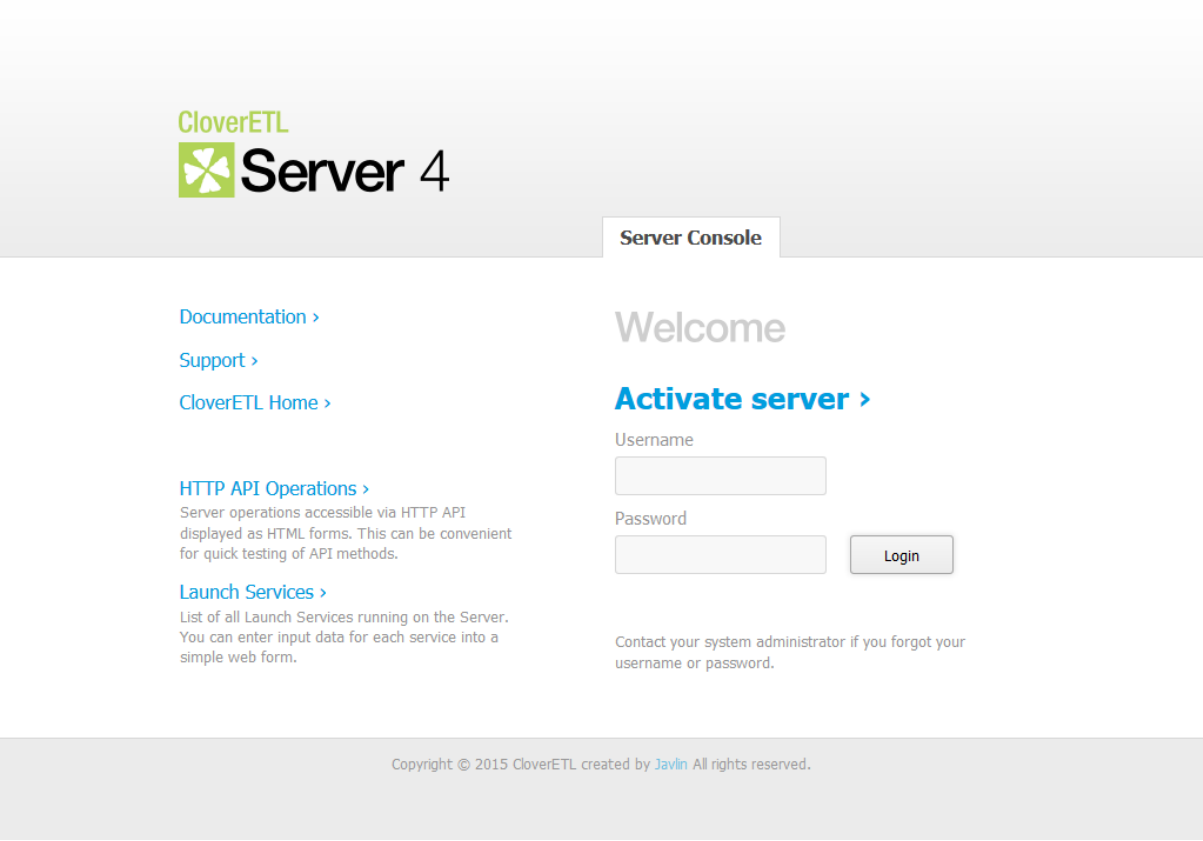


Figure 3.2. Login page of CloverETL Server without license

You can paste license text into **License key** or use **Browse** button to search for license file in the filesystem.

After clicking **Update** button the license is validated and saved to the database table *clover_licenses*. If the license is valid, a table with license's description appears. To proceed to CloverETL Serve console click **Continue to server console**.

To skip adding a license you can use **Close** button.

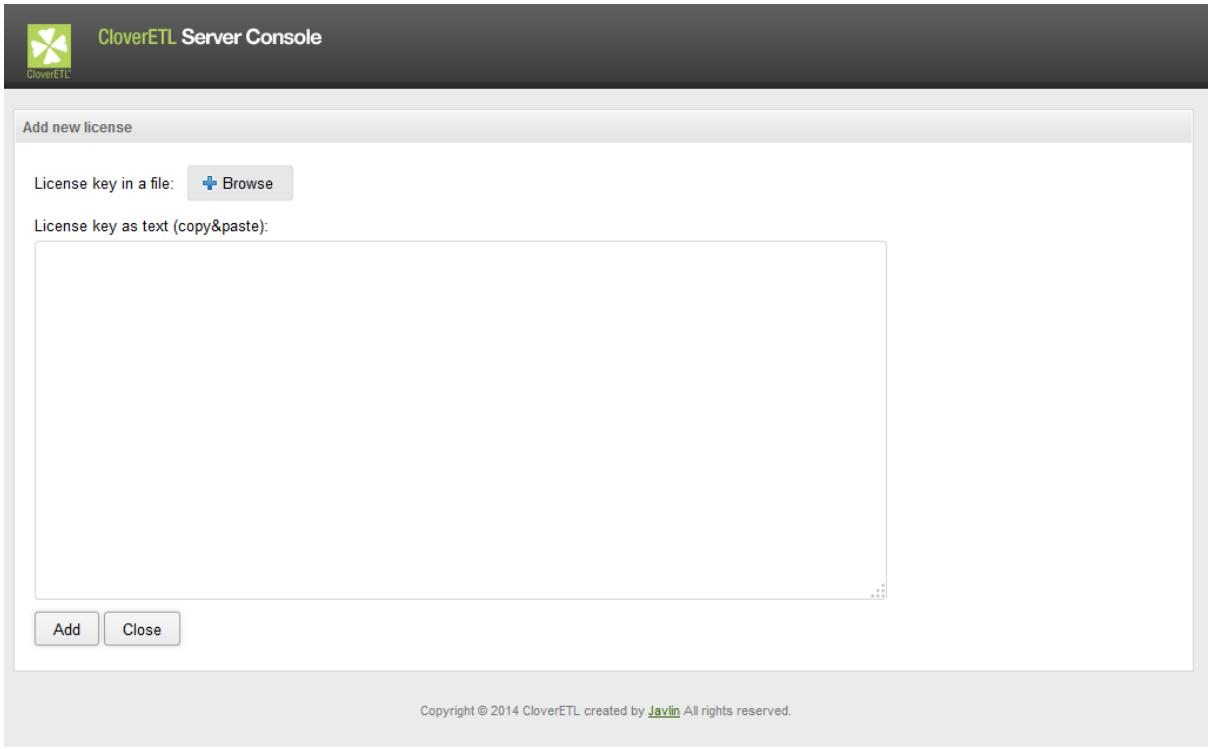


Figure 3.3. Add new license form

Update of CloverETL Server License in the Configuration section

If the license has been already installed, you can still change it by using form in the server web gui.

- Go to **server web GUI →Configuration →CloverETL Info →License**
- Click **Update license**.

You can paste license text into **License key** or use **Browse** button to search for license file in the filesystem. To skip adding a license you can use **Close** button.

After clicking **Update** button the license is saved to the database table *clover_licenses* and reloaded.

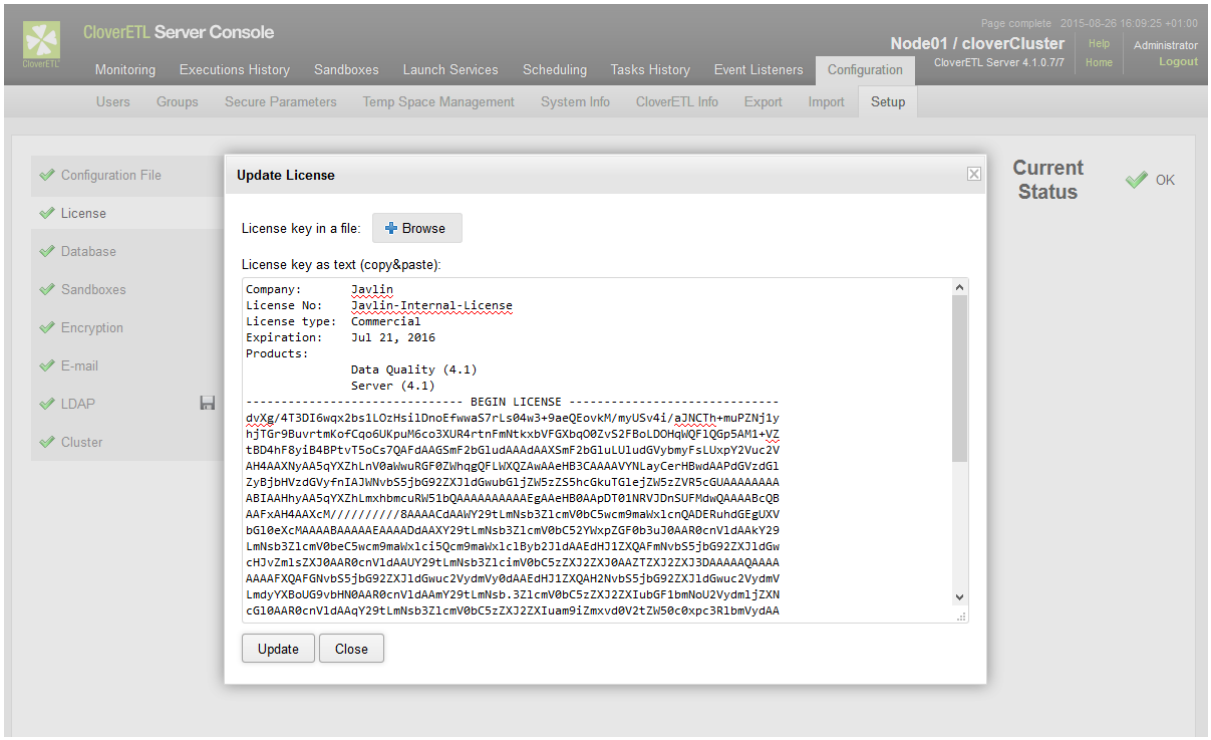


Figure 3.4. Update license form

Note

License in the database is common for all nodes in the cluster. Reloading of license occurs on each node in the cluster.

Note

If you assign more valid licenses, the most recent one is used.

Installation of CloverETL Server License using a license.file property

1. Get the `license.dat` file.
2. Set the CloverETL Server `license.file` parameter to the path to `license.dat`. Set its value to full path to the `license.dat` file.

   See Chapter 9, List of Properties (p. 65) for list of properties.
3. Restart the application server.

Separate License WAR

Simple approach, but it may be used only for standalone server running on Apache Tomcat.

1. Download the web archive file `clover-license.war`.
2. Copy `clover-license.war` to the `[tomcat_home]/webapps` directory.
3. The war file should be detected and deployed automatically without restarting Tomcat.
4. Check whether the license web-app is running on:

`http://[host]:[port]/clover-license/` (Note: contextPath `clover-license` is mandatory and cannot by changed)

Virtual MDM Plugin Installation

Downloading

Virtual MDM Components for **CloverETL Server** are downloaded as a ZIP file containing the extension. The ZIP file is available for download under your account on www.cloveretl.com in **CloverETL Server** download area, under the **Utilities** section as `virtual-mdm.zip` file.

Requirements

Requirements of **Virtual MDM Components**:

- supported OS are Microsoft Windows 32 bit, Microsoft Windows 64 bit, Linux 32 bit, Linux 64bit, and Mac OS X Cocoa
- at least 512MB of RAM
- installed CloverETL Server

Installation into Server

The following steps are needed to install **Virtual MDM Components** into **CloverETL Server**:

1. Install **CloverETL Server**, see its documentation for details.
2. Download the ZIP file with **Virtual MDM Components** for Server and store it on the system where **CloverETL Server** is installed. See Downloading (p. 31) for instructions for the download.
3. The ZIP file contains a **CloverETL** plugin. Your Server installation needs to be configured to find and load the plugin from the ZIP file. This is done by setting the `engine.plugins.additional.src` Server configuration property to the absolute path of the ZIP file, e.g. `engine.plugins.additional.src=c:/Server/virtual-mdm.4.1.0.zip` (in case the Server is configured via a property file).

   Details for setting the configuration property depend on your Server installation specifics, application server used etc. See **CloverETL Server** documentation for details. Typically the property would be set similarly to how you set-up the properties for connection to the Server's database. Updating the configuration property usually requires restart of the Server.
4. To verify that the plugin was loaded successfully, login to the Server's **Reporting Console** and look in the **Configuration** > **CloverETL Info** > **Plugins** page. In the list of plugins you should see `cloveretl.engine.initiate`.

Troubleshooting

If you get an `Unknown component` or `Unknown connection` error when running a graph with Virtual MDM components, it means that the **Virtual MDM Components** plugin was not loaded by the Server successfully. Please check the above steps to install the plugin, especially the path to the ZIP file.

Possible issues during installation

Since CloverETL Server is considered a universal JEE application running on various application servers, databases and jvm implementations, problems may occur during the installation. These can be solved by a proper configuration of the server environment. This section contains tips for the configuration.

Memory issues on Derby

If your server suddenly starts consuming too much resources (CPU, memory) despite having been working well before, it might be beacuse of running the internal Derby DB. Typically, causes are incorrect/incomplete shutdown of Apache Tomcat and parallel (re)start of Apache Tomcat.

Solution: move to a standard (standalone) database.

How to fix this? Redeploy CloverETL Server:

1. Stop Apache Tomcat and verify there are no other instances running. If so, kill them.
2. Backup config file, if you configured any.
3. Delete the `webapps/clover` directory.
4. Start Apache Tomcat server. It will automatically redeploy Clover Server.
5. Verify you can connect from Designer and from web.
6. Shutdown Apache Tomcat.
7. Restore config file and point it to your regular database.
8. Start Apache Tomcat.

JAVA_HOME or JRE_HOME environment variables are not defined

If you are getting this error message during an attempt to start your application server (mostly Tomcat), perform the following actions.

Linux:

These two commands will help you set paths to the variables on the server.

- `[root@server /] export JAVA_HOME=/usr/local/java`
- `[root@server /] export JRE_HOME=/usr/local/jdk`

As a final step, restart the application server.

Windows OS:

Set `JAVA_HOME` to your JDK installation directory, e.g. `C:\Program Files\java\jdk1.7.0`. Optionally, set also `JRE_HOME` to the JRE base directory, e.g. `C:\Program Files\java\jre7`.

Important

If you only have JRE installed, specify only `JRE_HOME`.

Apache Tomcat Context Parameters don't have any effect

Tomcat may sometimes ignore some of context parameters. It may cause wierd CloverETL Server behaviour, since it looks like configured, but only partially. Some parameters are accepted, some are ignored. Usually it works fine, however it may occur in some environments. Such behaviour is consistent, so after restart it's the same. It's probably related to Tomcat issues: https://issues.apache.org/bugzilla/show_bug.cgi?id=47516 and https://issues.apache.org/bugzilla/show_bug.cgi?id=50700 To avoid this, please use properties file instead of context parameters to configure CloverETL Server.

Tomcat log file catalina.out is missing on Windows

Tomcat start batch files for Windows aren't configured to create catalina.out file which contains standard output of the application. Catalinal.out may be vital when the Tomcat isn't started in console and any issue occurs. Or even when Tomcat is executed in the console, it may be closed automatically just after the error message appears in it.

Please follow these steps to enable catalina.out creation:

- Modify [tomcat_home]/bin/catalina.bat. Add parameter "/B" to lines where "_EXECJAVA" variable is set. There should be two these lines. So they will look like this:

  `set _EXECJAVA=start /B [the rest of the line]`
  Parameter /B causes, that "start" command doesn't open new console window, but runs the command it's own console window.

- Create new startup file. e.g. [tomcat_home]/bin/startupLog.bat, containing only one line:

  `catalina.bat start > ..\logs\catalina.out 2<&1`
  It executes Tomcat in the usual way, but standard output isn't put to the console, but to the catalina.out file.

Then use new startup file instead of [tomcat_home]/bin/startup.bat

Timeouts waiting for JVM

If you get the Jetty application server successfully running but cannot start Clover Server, it might be because of the wrapper waiting for JVM too long (it is considered a low-memory issue). Examine `[JETTY_HOME]\logs\jetty-service.log` for a line like this:

```
Startup failed: Timed out waiting for signal from JVM.
```

If it is there, edit `[JETTY_HOME]\bin\jetty-service.conf` and add these lines:

```
wrapper.startup.timeout=60
wrapper.shutdown.timeout=60
```

If that does not help either, try setting 120 for both values. Default timeouts are 30 both.

clover.war as default context on WebSphere (Windows OS)

If you are deploying `clover.war` on the IBM WebSphere server without context path specified, be sure to check whether it is the only application running in the context root. If you cannot start Clover Server on WebSphere, check the log and look for a message like this:

```
com.ibm.ws.webcontainer.exception.WebAppNotLoadedException:
Failed to load webapp: Failed to load webapp: Context root /* is already bound.
Cannot start application CloverETL
```

If you can see it, then this is the case. Getting rid of the issue, the easiest way is to stop all other (sample) applications and leave only `clover.war` running on the server. That should guarantee the server will be available in the context root from now on (e.g. http://localhost:9080/).

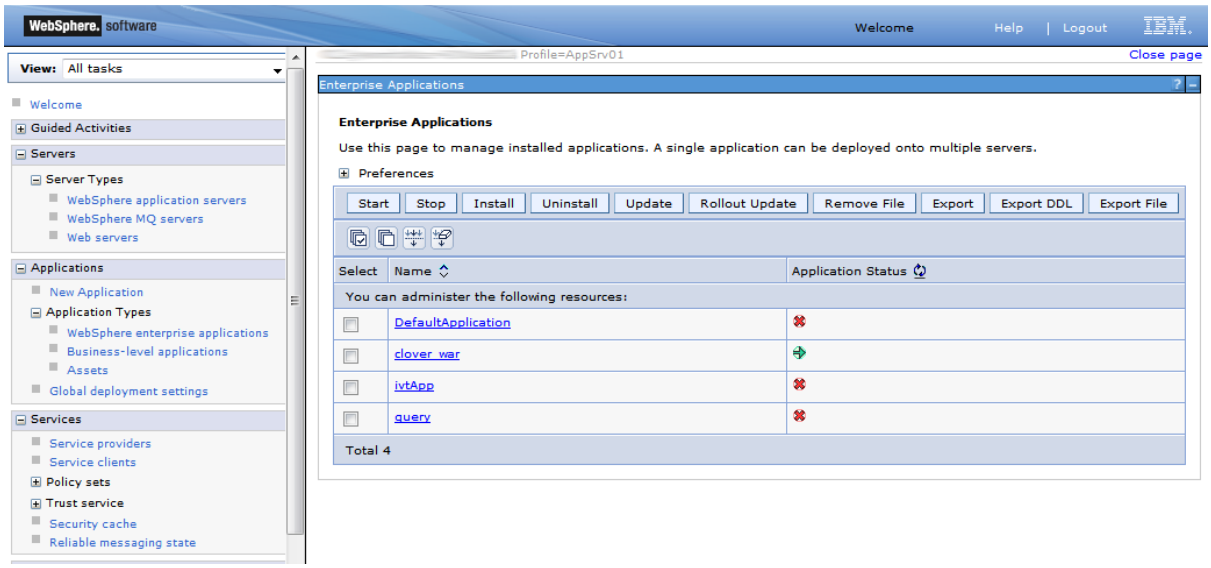


Figure 3.5. Clover Server as the only running application on IBM WebSphere

Tomcat 6.0 on Linux - Default DB

When using the internal (default) database on Linux, your Clover Server might fail on first start for no obvious reasons. Chances are that the `/var/lib/tomcat6/databases` directory was not created (because of access rights in parent folders).

Solution: Create the directory yourself and try restarting the server. This simple fix was successfully tested with Clover Server deployed as a `WAR` file via Tomcat web admin.

Derby.system.home cannot be accessed

If the server cannot start and the following message is in the log:

```
java.sql.SQLException: Failed to start database 'databases/cloverserver'
```

then see the next exception for details. After that check settings of the `derby.system.home` system property. It may point to an unaccessible directory, or files may be locked by another process. We suggest you set a specific directory as the system property.

Environment variables and more than one CloverETL Server instances running on single machine

If you are setting environment variables like `clover_license_file` or `clover_config_file` , remember you should not be running more than one CloverETL Server. Therefore if you ever needed to run more instances at once, use other ways of setting parameters (see Part III, "Configuration" (p. 40) for description of all possibilities) The reason is the environment variable is shared by all applications in use causing them to share configurations and fail unexpectedly. Instead of environment variables you can use system properties (passed to the application container process using parameter with -D prefix: -Dclover_config_file).

Special characters and slahes in path

When working with servers, you ought to stick to folder naming rules more than ever. Do not use any special characters in the server path, e.g. spaces, accents, diacritics are all not recommended. It's unfortunatelly common naming strategy on Windows systems. It can produce issues which are hard to find. If you are experiencing weird errors and cannot trace the source of them, why not install your application server in a safe destination like:

```
C:\JBoss6\
```

Similarly, use slashes but never backslahes in paths inside the `*.properties` files, e.g. when pointing to the Clover Server license file. If you incorrectly use backlash, it will be considered an escape character and the server may not work fine. This is an example of a correct path:

```
license.file=C:/CoverETL/Server/license.dat
```

File system permissions

Application server must be executed by OS user which has proper read/write permissions on file system. Problem may occur, if app-server is executed by root user for the first time, so log and other temp files are created by root user. When the same app-server is executed by another user, it will fail because it cannot write to root's files.

JMS API and JMS third-party libraries

Missing JMS libraries do not cause fail of server startup, but it is issue of deployment on application server, thus it still suits to this chapter.

clover.war itself does not contain jms.jar, thus it has to be on application server's classpath. Most of the application servers have jms.jar by default, but e.g. Tomcat does not. So if the JMS features are needed, the jms.jar has to be added explicitly.

If "JMS Task" feature is used, there must be third-party libraries on server's classpath as well. The same approach is recommended for JMS Reader/Writer components, even if these components allow to specify external libraries. It is due to common memory leak in these libraries which causes "OutOfMemoryError: PermGen space".

Using an unsupported JDBC connector for MySQL

CloverETL Server requires MySQL 5 up to version 5.5 included, using an unsupported JDBC connector for MySQL might cause exception, for example:

```
could not execute query
 You have an error in your SQL syntax; check the manual that coresponds to your MySQL server version for the right
    to use near 'OPTION SQL_SELECT_LIMIT=DEFAULT' at line 1
```

Chapter 4. Postinstallation configuration

Memory Settings

Current implementation of Java Virtual Machine allows only global configuration of memory for the JVM system process. Thus whole application server, together with WARs and EARs running on it, share one memory space.

Default JVM memory settings is too low for running application container with CloverETL Server. Some application servers, like IBM WebSphere, increase JVM defaults themselves, however they still may be too low.

The best memory limits depend on many conditions, i.e. transformations which CloverETL should execute. Please note, that maximum limit isn't amount of permanently allocated memory, but limit which can't be exceeded. If the limit was exhaused, the OutOfMemoryError would be raised.

You can set the minimum and maximum memory heap size by adjusting the "Xms" and "Xmx" JVM parameters. There are more ways how to change the settings depending on the used application container.

Moreover JVM doesn't use just HEAP memory, but also PermGen space(for loaded classes), so called direct memory(used by graph edges) and stack memory(allocated for each thread). Thus it's not wise to set HEAP limit too high, since if it consumed whole RAM, JVM wouldn't be able to allocate direct memory and stack for new threads.

If you have no idea about the memory required for the transformations, a maximum of 1-2 GB heap memory is recommended. This limit may be increased during transformations development when `OutOfMemoryError` occurs.

Memory space for loading classes (so called "PermGen space") is separated from heap memory, and can be set by the JVM parameter "-XX:MaxPermSize". By default, it is just 64 MB which is not enough for enterprise applications. Again, suitable memory limit depends on various criteria, but 512 MB should be enought in most cases. If the PermGen space maximum is too low, `OutOfMemoryError: PermGen space` may occur.

Please see the specific container section for details how to make the settings.

Maximum Number of Open Files

When using resource-greedy components, such as FastSort, or when running a large number of graphs concurrently, you may hit the system limit of simultaneously open files. This is usually indicated by the `java.io.IOException: Too many open files` exception.

The default limit is fairly low in many Linux distributions (e.g. 4096 in Ubuntu). Such a limit can be easily exceeded, considering that one FastSort component can open up to 1,000 files when sorting 10 million records. Furthermore, some application containers recommend increasing the limit themselves (8192 for IBM WebSphere).

Therefore it is recommended to increase the limit for production systems. Reasonable limits vary from 10,000 to about 100,000 depending on the expected load of **CloverETL Server** and the complexity of your graphs.

The current limit can be displayed in most UNIX-like systems using the **ulimit -Hn** command.

The exact way of increasing the limit is OS-specific and is beyond the scope of this manual.

Chapter 5. Upgrading Server to Newer Version

General Notes on Upgrade

- Upgrade of CloverETL Server requires a down time; plan maintenance window
- Succesful upgrade requires about 30 minutes; rollback requires 30 minutes
- Performing the below steps in development/testing environment first before moving onto production one

Upgrade Prerequisites

- Having new CloverETL Server web application archive (`clover.war` appropriate for the application server used) & license files available
- Having release notes for particular CloverETL version available (and all version between current and intended version to be upgraded to)
- Having the graphs and jobs updated and tested with regards to Known Issues & Compatibility for particular CloverETL version.
- Having the CloverETL Server configuration properties file externalized from default location, see Chapter 6, Config Sources and Their Priorities (p. 41)
- Standalone database schema where CloverETL Server stores configuration, see Chapter 8, Examples of DB Connection Configuration (p. 51)
- Having a separate sandbox with test graph that can be run anytime to verify that CloverETL Server runs correctly and allows for running jobs

Upgrade Instructions

1. Suspend all sandboxes, wait for running graphs to finish processing
2. Shutdown CloverETL Server application (or all servers, if they run in cluster mode)
3. Backup existing CloverETL database schema (if any changes to the database schema are necessary, the new server will automatically make them when you start it for the first time)
4. Backup existing CloverETL web application archive (`clover.war`) & license files (on all nodes)
5. Backup existing CloverETL sandboxes (on all nodes)
6. Re-deploy the CloverETL Server web application. Instructions how to do that are application server dependant - see Enterprise Server (p. 10) for installation details on all supported application servers. After you re-deploy, your new server will be configured based on the previous version's configuration.
7. Replace old license files by the valid one (or you can later use web GUI form to upload new license). The license file is shipped as a text containing a unique set of characters. If you:
   - received the new license as a file (`*.dat`), then simply use it as new license file.
   - have been sent the licence text e.g inside an e-mail, then copy the license contents (i.e. all text between `Company` and `END LICENSE`) into a new file called `clover-license.dat`. Next, overwrite the old license file with the new one or upload it in the web GUI.

   See Installation of CloverETL Server License (p. 27) for deatails on license installation.
8. Start CloverETL Server application (on all nodes)

9. Review that contents of all tabs in CloverETL Server Console, especially scheduling and event listeners looks OK.

10. Update graphs to be compatible with the particular version of CloverETL Server (this should be prepared and tested in advance)

11. Resume the test sandbox and run a test graph to verify functionality

12. Resume all sandboxes

Rollback Instructions

1. Shutdown CloverETL Server application

2. Restore CloverETL Server web application (`clover.war`) & license files (on all nodes)

3. Restore CloverETL Server database schema

4. Restore CloverETL sandboxes (on all nodes)

5. Start CloverETL Server application (on all nodes)

6. Resume the test sandbox and run a test graph to verify functionality

7. Resume all sandboxes

Important

Evaluation Version - a mere upgrade of your license is not sufficient. When moving from evaluation to enterprise server, you should not use the default configuration and database. Instead, take some time to configure Clover Server so that it best fits your production environment.

Part III. Configuration

We recommend the default installation (without any configuration) only for evaluation purposes. For production use, we recommend configuring a dedicated database and properly configuring the SMTP server for sending notifications.

Chapter 6. Config Sources and Their Priorities

There are several sources of configuration properties. If property isn't set, application default is used.

Configuration properties can be encrypted (see details in Chapter 10, Secure configuration properties (p. 70)).

Warning: Do not combine sources specified below. Configuration becomes confusing and maintenance will be much more difficult.

Environment Variables

Set environment variable with prefix `clover.` , i.e. (`clover.config.file`)

Some operating systems may not use dot character, so also underlines (_) may be used instead of dots (`.`). So the `clover_config_file` works as well.

System Properties

Set system property with prefix `clover.` , i.e. (`clover.config.file`)

Also underlines (_) may be used instead of dots (`.`) so the `clover_config_file` works as well.

Properties File on default Location

Source is common properties file (text file with key-value pairs):

```
[property-key]=[property-value]
```

By default CloverETL tries to find config file `[workingDir]/cloverServer.properties`.

Properties File on specified Location

The same as above, but properties file is not loaded from default location, because its location is specified by environment variable or system property `clover_config_file` or `clover.config.file`. This is recommended way of configuration if context parameters cannot be set in application server.

Modification of Context Parameters in web.xml

Unzip `clover.war` and modify file `WEB-INF/web.xml`, add this code:

```
<context-param>
  <param-name>[property-name]</param-name>
  <param-value>[property-value]</param-value>
</context-param>
```

This way isn't recommended, but it may be useful when none of above ways is possible.

Context Parameters (Available on Apache Tomcat)

Some application servers allow to set context parameters without modification of WAR file.

This way of configuration is possible, but Apache Tomcat may ignore some of context parameters in some environments, so this way isn't recommended, use of properties file is almost as convenient and much more reliable way.

Example for Apache Tomcat

On Tomcat it is possible to specify context parameters in context configuration file. `[tomcat_home]/conf/Catalina/localhost/clover.xml` which is created automatically just after deployment of CloverETL Server web application.

You can specify property by adding this element:

```
<Parameter name="[propertyName]" value="[propertyValue]" override="false" />
```

Priorities of config Sources

Configuration sources have these priorities:

1. context parameters (specified in application server or directly in `web.xml`)
2. external config file CS tries to find it in this order (only one of them is loaded):
   - path specified by context parameter `config.file`
   - path specified by system property `clover_config_file` or `clover.config.file`
   - path specified by environment variable `clover_config_file` or `clover.config.file`
   - default location (`[workingDir]/cloverServer.properties`)
3. system properties
4. environment variables
5. default values

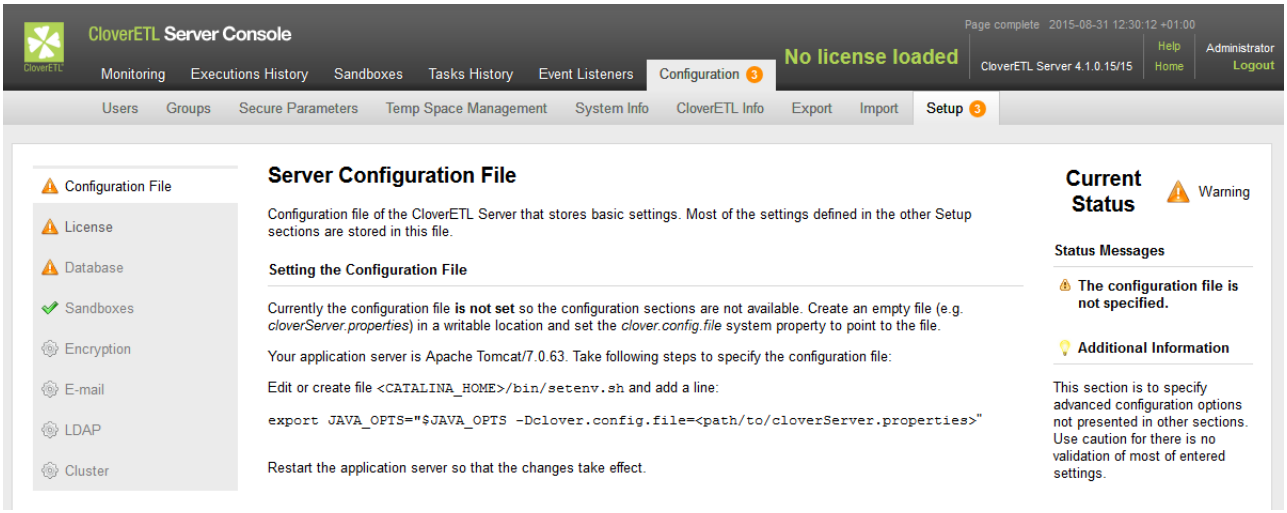
Chapter 7. Setup

CloverETL Server Setup helps you with configuration of CloverETL server. Instead of typing the whole configuration file in text editor, the **Setup** generates content of the configuration file according to your instructions. It let you set up **License** and configure **Database Connection**, **LDAP Connection**, **SMTP Server Connection**, **Sandbox Paths**, **Encryption** and **Cluster Configuration**.

The Setup is accessible from **Server Console** under **Configuration** →**Setup**.

Using Setup

If you start an unconfigured server, you can see decorators pointing to Setup. The decorators mark problems to be solved. The displayed number corresponds to the number of items.



The following states mean error as mentioned in the text above:

- error
- warning
- restart required

Setup will help you with solving the problems.

Path to Configuration File

Firstly, you have to configure path to configuration file. Without this, the Setup does not know, to which file the configuration should be saved. Each application server has different way to configure it.

Apache Tomcat

Edit `bin/setenv.sh` (or `bin/setenv.bat`) and add `-Dclover.config.file=/absolute/path/to/cloverServer.properties` to `CATALINA_OPTS`.

See also Apache Tomcat (p. 10).

Jetty

Edit `bin/jetty.sh` and add `-Dclover.config.file=/absolute/path/to/cloverServer.properties` to `JAVA_OPTS`.

See also Jetty (p. 14).

Glassfish

Add `clover.config.file` property in application server GUI (accessible on http://localhost:4848). The property can be added under **Configuration →System Properties**.

See also Glassfish / Sun Java System Application Server (p. 18).

JBoss

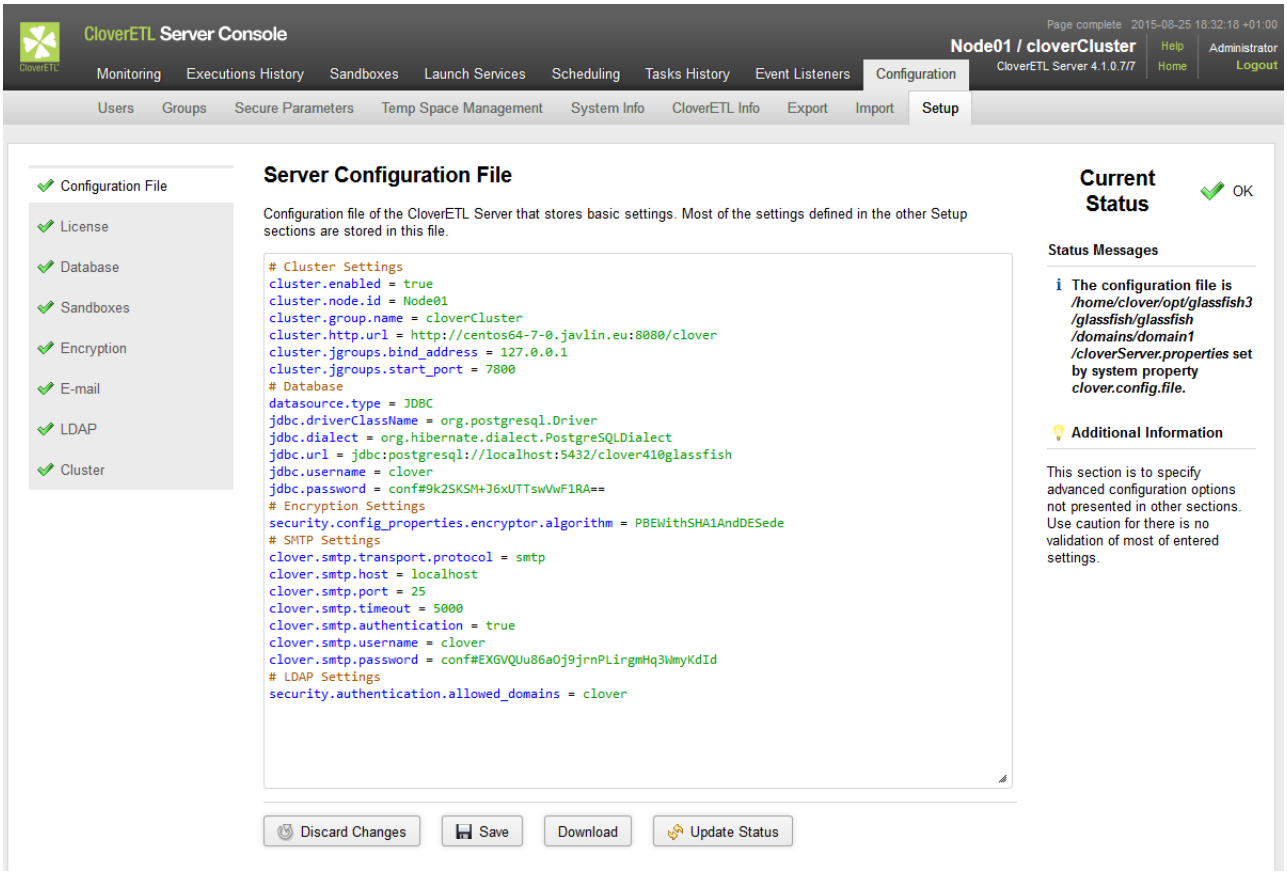
See also JBoss Application Server (p. 19).

Websphere

See also IBM WebSphere (p. 15).

Weblogic

See also Oracle WebLogic Server (p. 25).



Adding Libraries to Classpath

Secondly, you should configure connection to database.

Place necessary libraries to suitable directory. You usually need jdbc driver for connection to database or .jar file with encryption provider.

Having added the libraries restart the application server and configure Clover Server using Setup.

Configuring Particular Items

Use **Setup**. Items configured in **Setup** are saved into file defined by `clover.config.file`.

If you need encryption, configure the **Encryption** first.

Configure connection to database and then update license. Later, you can configure other setup items.

Some **Setup** items require restart of application server (Database and Cluster). The latter items (License, Sandboxes, E-mail, LDAP) do not need restarting; the changes to them are applied immediately. If you change something in **Configuration file** tab, restart it or not depending on updated part of the file.

As the last step, restart the server to use the new configuration.

Setup Tabs

Each setup page consists of menu with setup tabs on the left, main configuration part in the middle and configuration status and text on the right side.

Menu tabs have icons surrounding the text: tick marks configured tab, wheel marks inactive tab, floopy disk maks configuration that needs saving. Arrows signalize request on restart.

The main configuration part contains several buttons:

Discard Changes discards unsaved changes and returns to rurrently used values.

Save checks the configuration. If the configuration is valid, it is saved to the configuration file. If the confuguration is not valid, **Save Anyway** button appears. The **Save Anyway** button allows you to save configuration considered as invalid. E.g. Database connection is considered invalid if there is a required library missing. If you see **Save** disabled, use **Validate** to validate configuration first.

Validate validates the configuration on current tab.

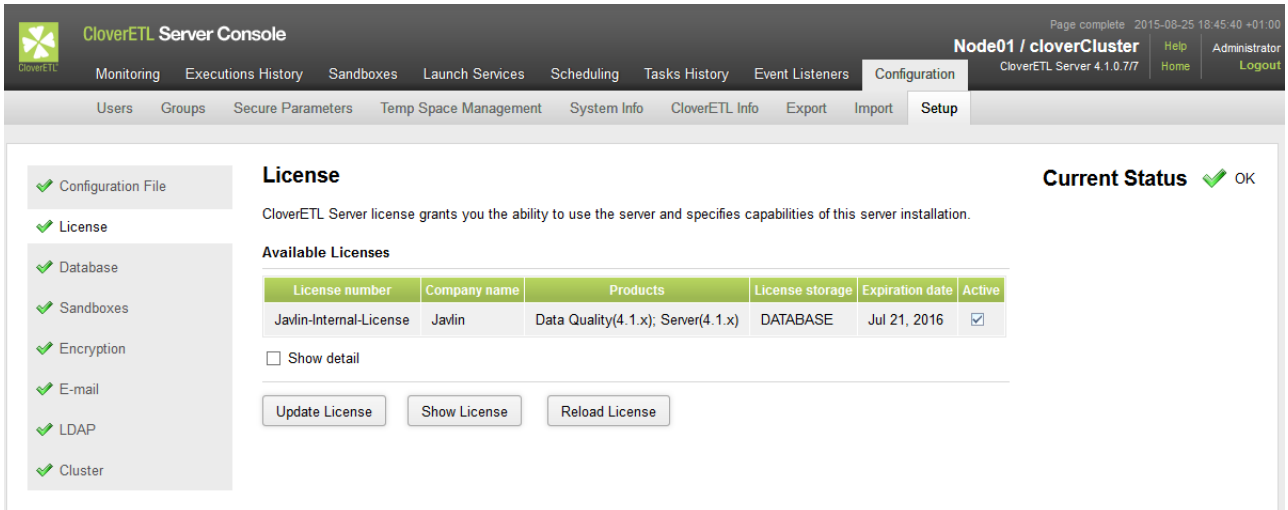
Configuration File

Configuration tab displays content of configuration file. You do not have to edit the content of the file manually, use particular tab to configure corresponding subsystem.

License

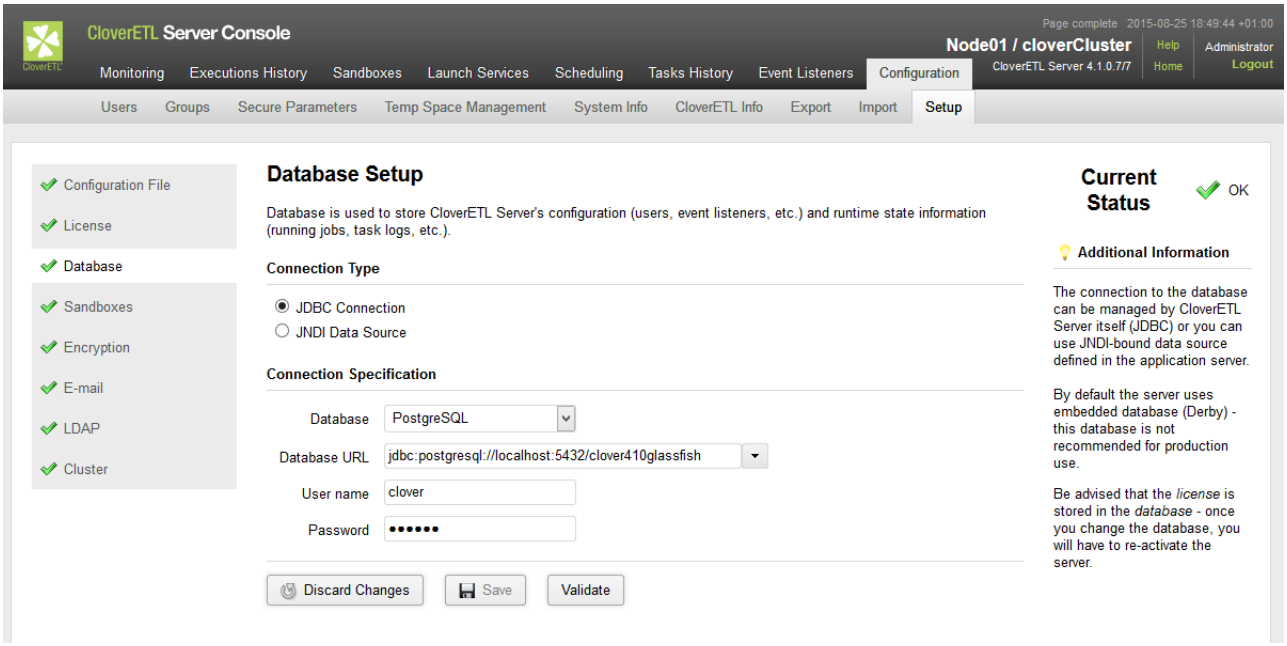
License tab let you specify license. The license is stored in database.

You should configure database before specifying license. Otherwise, you will have to specify the license twice.

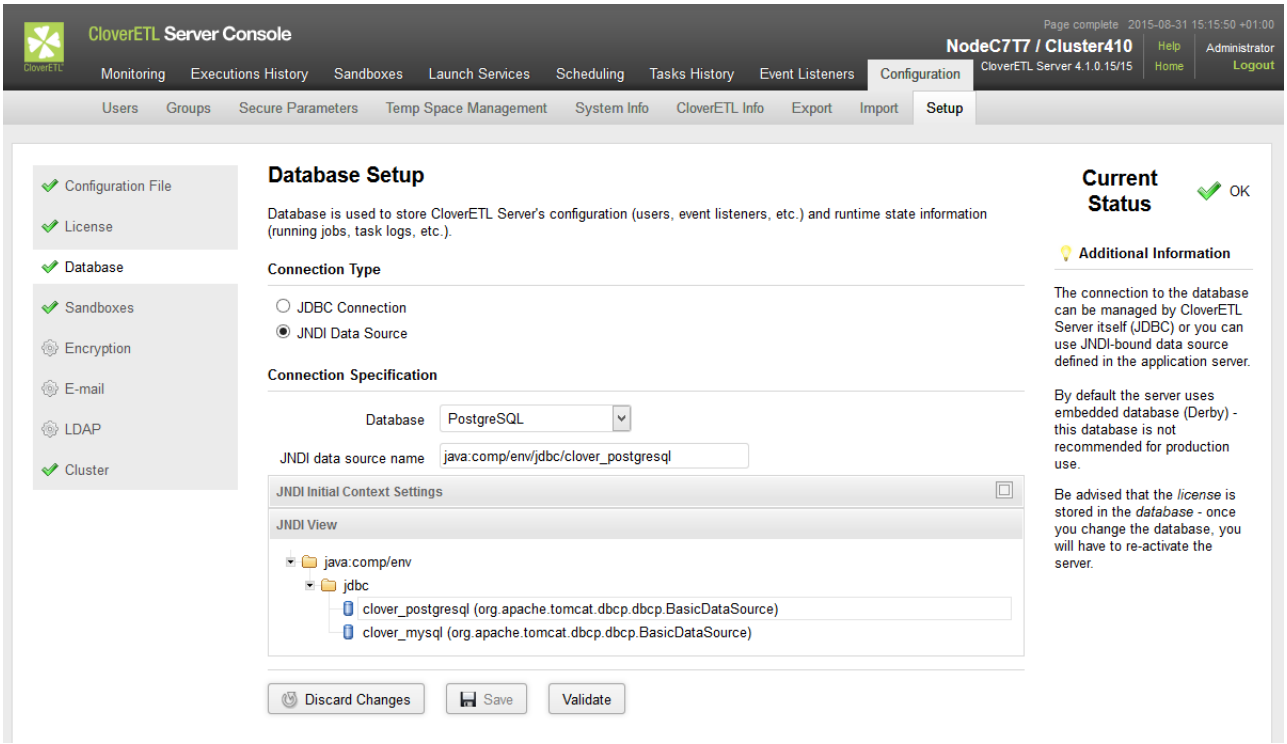


Database

Database tab let you configure connection to database. You can connect via JDBC.

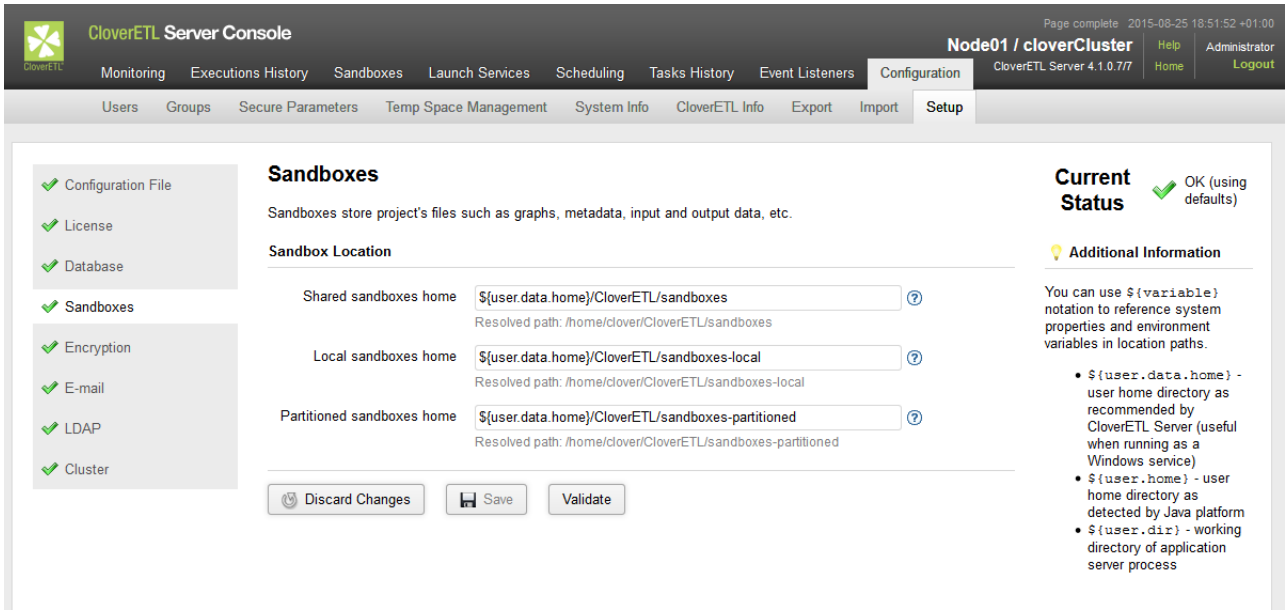


Or you can use JNDI to access the datasource on application server level. Choose a suitable item of JNDI tree.



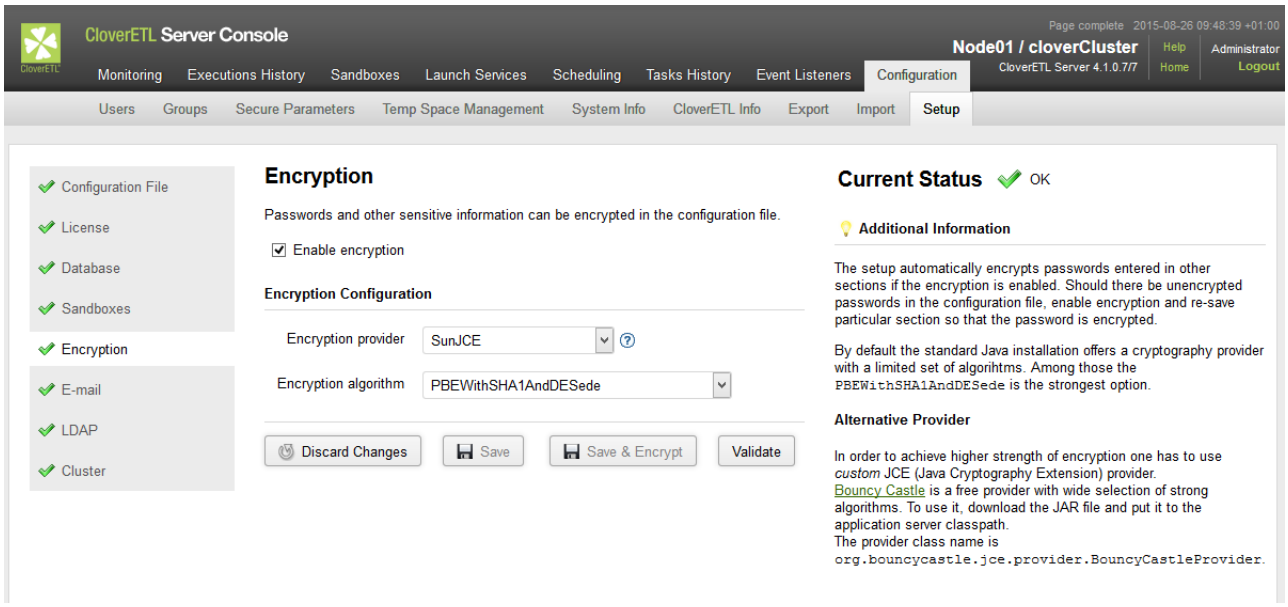
Sandboxes

Sandboxes let you configure path to sandboxes: shared, local, partitioned.



Encryption

Encryption tab let you enable encryption of sensitive items of configuration file. You can choose encryption provider and encryption algorithm. Alternative encryption provider can be used; the libs have to be added on `classpath`. (In the same vay as database libraries.)

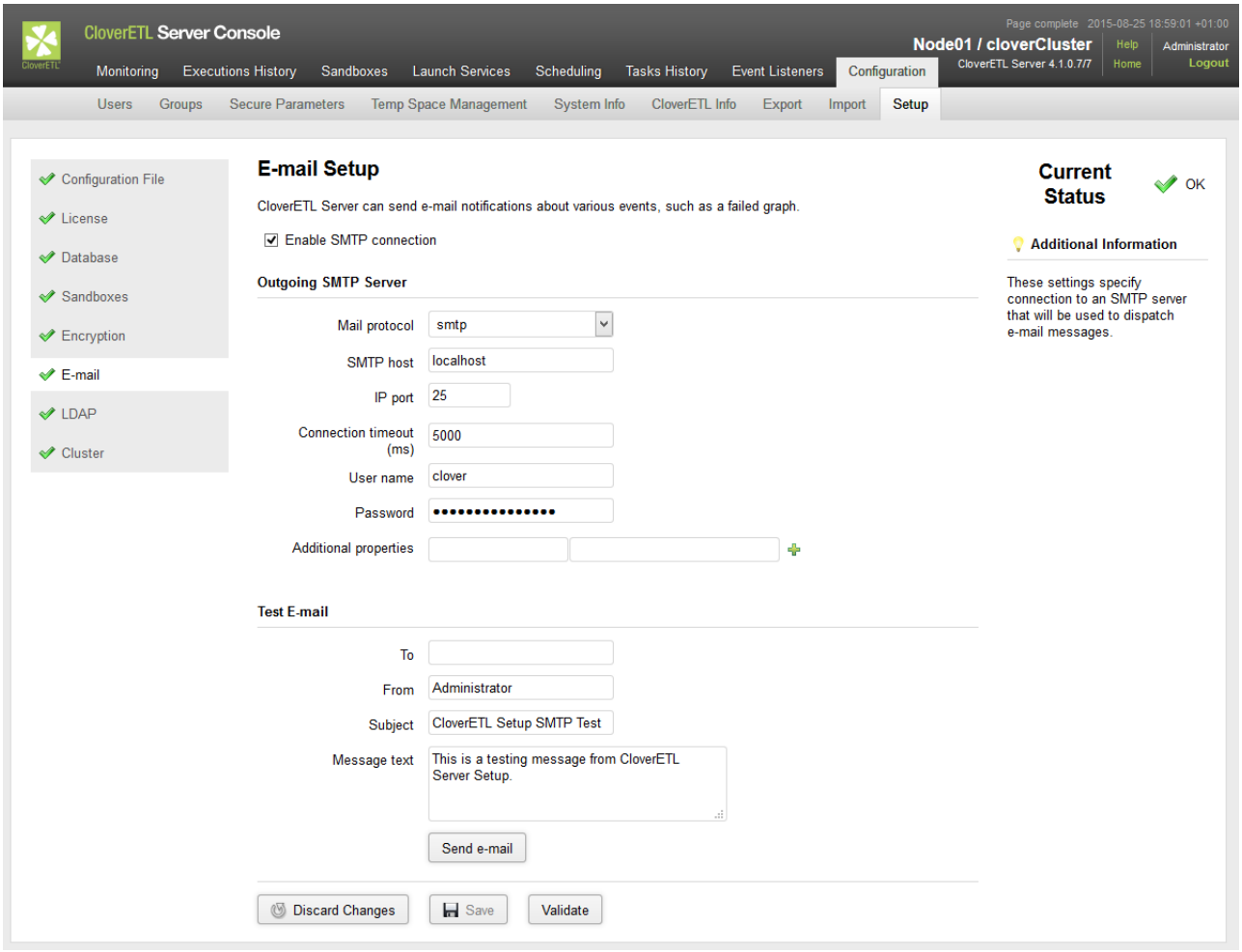


Save & Encrypt button saves configuration and encrypts the passwords.

E-Mail

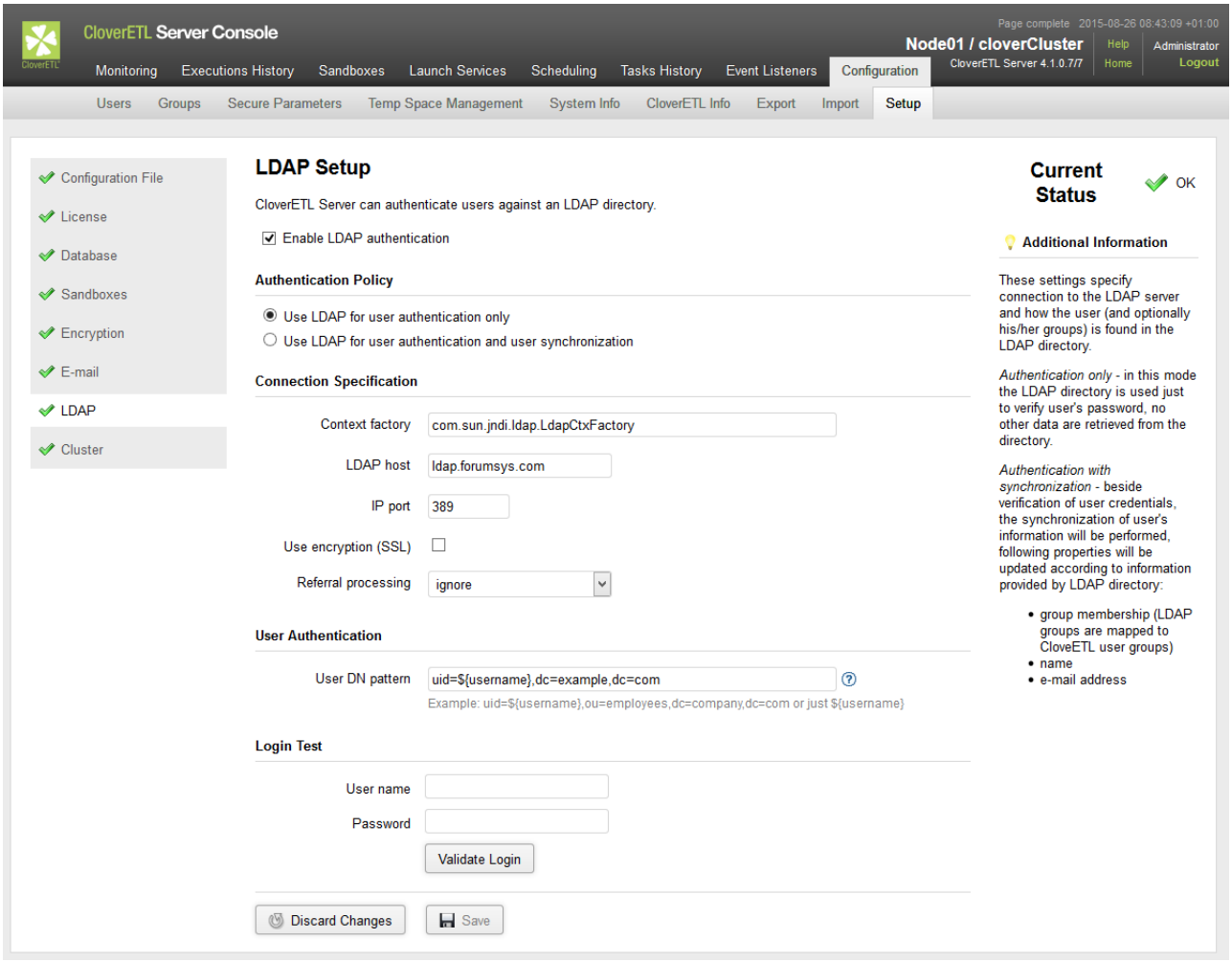
E-mail tab let you configure connection to SMTP server. The connection is necessary for reporting event on server via emails.

E-mail configuration can be tested with sending an e-mail from the dialog.



LDAP

LDAP tab let you use existing LDAP database for user authentication.

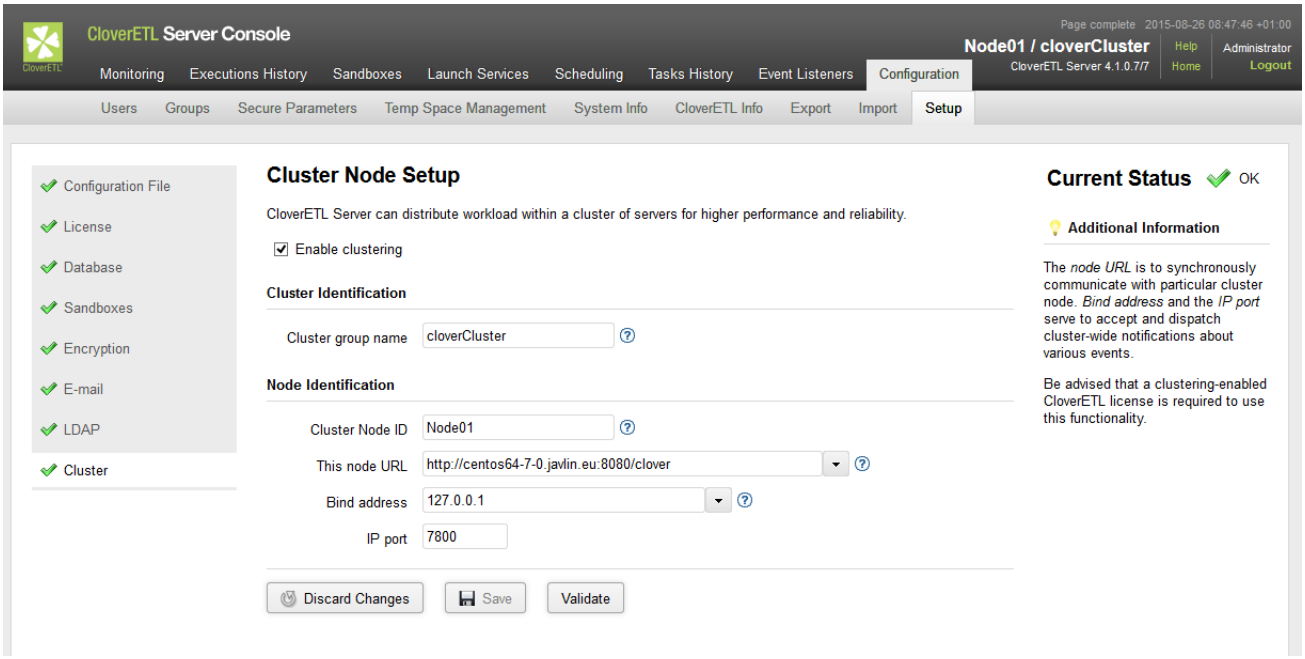


Firstly, you should specify connection to the LDAP server. Secondly, define pattern for user DN. The login can be validated using any user matching the pattern.

See also LDAP Authentication (p. 86).

Cluster

Cluster tab let you configure clustering features.



Note

You can use setup in a fresh installation of CloverETL Server, even if it had not been activated yet: log in into Server Console and use **Close** button to access the menu.

Chapter 8. Examples of DB Connection Configuration

In standalone deployment (non-clustered), configuration of DB connection is optional, since embedded Apache Derby DB is used by default and it is sufficient for evaluation. However, configuration of external DB connection is strongly recommended for production deployment. It is possible to specify common JDBC DB connection attributes (URL, username, password) or JNDI location of DB DataSource.

In clustered deployment, at least one node in cluster must have DB connection configured. Other nodes may have their own direct connection (to the same DB) or may use another node as proxy for persistent operations, however scheduler is active only on nodes with direct connection. See Part VI, "Cluster" (p. 182) for details about this feature, this section describes only direct DB connection configuration.

DB Configurations and their changes may be as follows:

See Configuration Repository (p. 6) for officialy supported versions of particular databases.

Embedded Apache Derby

Apache Derby embedded DB is used with default CloverETL Server installation. It uses working directory as storage directory for data persistence by default. This may be problem on some systems. In case of any problems with connecting to Derby DB, we recommend you configure connection to external DB or at least specify Derby home directory:

Set system property `derby.system.home` to set path which is accessible for application server. You can specify this system property by this JVM execution parameter:

```
-Dderby.system.home=[derby_DB_files_root]
```

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName`, `jdbc.url`, `jdbc.username`, `jdbc.password`, `jdbc.dialect`. For example:

```
jdbc.driverClassName=org.apache.derby.jdbc.EmbeddedDriver
jdbc.url=jdbc:derby:databases/cloverDb;create=true
jdbc.username=
jdbc.password=
jdbc.dialect=com.cloveretl.server.dbschema.DerbyDialect
```

Take a closer look at the `jdbc.url` parameter. Part `databases/cloverDb` means a subdirectory for DB data. This subdirectory will be created in the directory which is set as `derby.system.home` (or in the working directory if `derby.system.home` is not set). Value `databases/cloverDb` is a default value, you may change it.

Derby JDBC 4 compliant driver is bundled with CloverETL Server, thus there is no need to add it on the classpath.

MySQL

CloverETL Server supports MySQL 5, up to version 5.5 included.

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName`, `jdbc.url`, `jdbc.username`, `jdbc.password`, `jdbc.dialect`. For example:

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://127.0.0.1:3306/clover?useUnicode=true&characterEncoding=utf8
jdbc.username=root
jdbc.password=
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

Create DB with proper charset, like this:

```
CREATE DATABASE IF NOT EXISTS clover DEFAULT CHARACTER SET 'utf8';
```

DB2

DB2 on Linux/Windows

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName`, `jdbc.url`, `jdbc.username`, `jdbc.password`, `jdbc.dialect`. For example:

```
jdbc.driverClassName=com.ibm.db2.jcc.DB2Driver
jdbc.url= jdbc:db2://localhost:50000/clover
jdbc.username=usr
jdbc.password=pwd
jdbc.dialect=org.hibernate.dialect.DB2Dialect
```

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

Possible problems

Wrong pagesize

Database *clover* has to be created with suitable `PAGESIZE`. DB2 has several possible values for this property: 4096, 8192, 16384 or 32768.

CloverETL Server should work on DB with `PAGESIZE` set to 16384 or 32768. If `PAGESIZE` value is not set properly, there should be error message in the log file after failed CloverETL Server startup:

```
ERROR:
DB2 SQL Error: SQLCODE=-286, SQLSTATE=42727, SQLERRMC=16384;
ROOT, DRIVER=3.50.152
```

SQLERRMC contains suitable value for `PAGESIZE`.

You can create database with proper `PAGESIZE` like this:

```
CREATE DB clover PAGESIZE 32768;
```

The table is in the reorg pending state

After some ALTER TABLE commands, some tables may be in "reorg pending state". This behaviour is specific for DB2. ALTER TABLE DDL commands are executed only during the first start of new CloverETL Server version.

Error message for this issue may look like this:

```
Operation not allowed for reason code "7" on table "DB2INST2.RUN_RECORD"..
        SQLCODE=-668, SQLSTATE=57016
```

or like this

```
DB2 SQL Error: SQLCODE=-668, SQLSTATE=57016, SQLERRMC=7;DB2INST2.RUN_RECORD, DRIVER=3.50.152
```

In this case "RUN_RECORD" is table name which is in "reorg pending state" and "DB2INST2" is DB instance name.

To solve this, go to DB2 console and execute command (for table run_record):

```
reorg table run_record
```

DB2 console output should look like this:

```
db2 => connect to clover1
Database Connection Information

Database server        = DB2/LINUX 9.7.0
SQL authorization ID   = DB2INST2
Local database alias   = CLOVER1

db2 => reorg table run_record
DB20000I  The REORG command completed successfully.
db2 => disconnect clover1
DB20000I  The SQL DISCONNECT command completed successfully.
```

"clover1" is DB name

DB2 does not allow ALTER TABLE which trims DB column length.

This problem depends on DB2 configuration and we've experienced this only on some AS400s so far. CloverETL Server applies set of DP patches during the first installation after application upgrade. Some of these patches may apply column modifications which trims length of the text columns. These changes never truncate any data, however DB2 does not allow this since it "may" truncate some data. DB2 refuses these changes even in DB table which is empty. Solution is, to disable the DB2 warning for data truncation, restart CloverETL Server which applies patches, then enable DB2 warning again.

DB2 on AS/400

The connection on AS/400 might be slightly different.

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName`, `jdbc.url`, `jdbc.username`, `jdbc.password`, `jdbc.dialect`. For example:

```
jdbc.driverClassName=com.ibm.as400.access.AS400JDBCDriver
jdbc.username=javlin
jdbc.password=clover
jdbc.url=jdbc:as400://host/cloversrv;libraries=cloversrv;date format=iso
jdbc.dialect=org.hibernate.dialect.DB2400Dialect
```

Use credentials of your OS user for `jdbc.username` and `jdbc.password`.

`cloversrv` in `jdbc.url` above is the name of the DB schema.

You can create schema in AS/400 console:

- execute command `STRSQL` (**SQL console**)
- execute `CREATE COLLECTION cloversrv IN ASP 1`
- `cloversrv` is the name of the DB schema and it may be at most 10 characters long

Proper JDBC driver must be in the application server classpath.

I use JDBC driver `jt400ntv.jar`, which I've found in `/QIBM/ProdData/Java400` on the server.

Use `jt400ntv.jar` JDBC driver.

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

Oracle

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName, jdbc.url, jdbc.username, jdbc.password, jdbc.dialect.` For example:

```
jdbc.driverClassName=oracle.jdbc.OracleDriver
jdbc.url=jdbc:oracle:thin:@host:1521:db
jdbc.username=user
jdbc.password=pass
jdbc.dialect=org.hibernate.dialect.Oracle10gDialect
```

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

These are privileges which have to be granted to schema used by CloverETL Server:

```
CONNECT
CREATE SESSION
CREATE/ALTER/DROP TABLE
CREATE/ALTER/DROP SEQUENCE

QUOTA UNLIMITED ON <user_tablespace>;
QUOTA UNLIMITED ON <temp_tablespace>;
```

MS SQL

MS SQL requires configuration of DB server.

- Allowing of TCP/IP connection:
- execute tool **SQL Server Configuration Manager**
- go to **Client protocols**
- switch on TCP/IP (default port is 1433)
- execute tool **SQL Server Management Studio**
- go to **Databases** and create DB clover
- go to **Security/Logins** and create user and assign this user as owner of DB *clover*
- go to **Security** and check **SQL server and Windows authentication mode**

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName, jdbc.url, jdbc.username, jdbc.password, jdbc.dialect.` For example:

```
jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
jdbc.url=jdbc:sqlserver://localhost:1433;databaseName=clover
jdbc.username=user
jdbc.password=pass
jdbc.dialect=org.hibernate.dialect.SQLServerDialect
```

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

Postgre SQL

If you use properties file for configuration, specify these parameters: `jdbc.driverClassName, jdbc.url, jdbc.username, jdbc.password, jdbc.dialect.` For example:

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost/clover?charSet=UTF-8
jdbc.username=postgres
jdbc.password=
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Please don't forget to add JDBC 4 compliant driver on the classpath. JDBC Driver which doesn't meet JDBC 4 won't work properly.

The JDBC driver for PostgreSQL can be downloaded from: https://jdbc.postgresql.org/download.html. In Apache Tomcat you would place libs into `$CATALINA_HOME/libs` directory.

JNDI DB DataSource

Server can connect to JNDI DB DataSource, which is configured in application server or container. However there are some CloverETL parameters which must be set, otherwise the behaviour may be unpredictable:

```
datasource.type=JNDI # type of datasource; must be set, because default value is JDBC
datasource.jndiName=# JNDI location of DB DataSource; default value is java:comp/env/jdbc/clover_server #
jdbc.dialect=# Set dialect according to DB which DataSource is connected to.
The same dialect as in sections above. #
```

The parameters above may be set in the same ways as other params (in properties file or Tomcat context file)

Example of DataSource configuration in Apache Tomcat. Add following code to context file (`webapps/clover/META-INF/context.xml`).

```
<Resource name="jdbc/clover_server" auth="Container"
    type="javax.sql.DataSource" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://192.168.1.100:3306/clover?useUnicode=true&amp;characterEncoding=utf8"
    username="root" password="" maxActive="20" maxIdle="10" maxWait="-1"/>
```

Note

Special characters you type in the context file have to be specified as XML entities. E.g. ampersand "&" as "&" etc.

See Chapter 9, List of Properties (p. 65) for list of properties.

Encrypted JNDI

You can store password for database connection encrypted. The configuration differs between particular application servers.

Encrypted JNDI on Tomcat

You need `secure-cfg-tool` to encrypt the passwords. Use version of `secure-cfg-tool` corresponding to version of CloverETL Server. Usage of the tool is described in Chapter 10, Secure configuration properties (p. 70).

Use `encrypt.sh` or `encrypt.bat` for encryption of a password. Place the encrypted password into configuration file and put files `cloveretl-secure-jndi-resource-{version}.jar` and `jasypt-1.9.0.jar` on classpath of application server. The `.jar` files can be found in `tomcat-secure-jndi-resource` directory packed in secure-cfg-tool.

The directory `tomcat-secure-jndi-resource` contains useful file `README` with further details on encrypted JNDI.

Example of encrypted JNDI connection for Postgresql

Encrypt the password:

```
./encrypt.sh -a PBEWithSHA1AndDESede
```

The configuration is placed in `${CATALINA_HOME}/webapps/clover/META-INF/cotext.xml`. Note that the encryption algorithm PBEWithSHA1AndDESede is not default.

```
<Resource name="jdbc/clover_server"
          auth="Container"
          factory="com.cloveretl.secure.tomcatresource.SecureDataSourceFactory"
          secureAlgorithm="PBEWithSHA1AndDESede"
          type="javax.sql.DataSource"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://127.0.0.1:5432/clover410m1?charSet=UTF-8"
          username="conf#rPz5Foo7HPn4dFTRV5Ourg=="
          password="conf#4KlNp8/FVDR+rTWX0dEqWA=="
          maxActive="20"
          maxIdle="10"
          maxWait="-1"/>
```

If you use other JCE (e.g. Bouncy Castle), it has to be added to classpath of application server (`${CATALINA_HOME}/lib`). The encrypt command requires path to directory with JCE too.

```
./encrypt.sh -l ~/lib/ -c org.bouncycastle.jce.provider.BouncyCastleProvider
-a PBEWITHSHA256AND256BITAES-CBC-BC
```

```
<Resource name="jdbc/clover_server"
          auth="Container"
          factory="com.cloveretl.secure.tomcatresource.SecureDataSourceFactory"
          secureProvider="org.bouncycastle.jce.provider.BouncyCastleProvider"
          secureAlgorithm="PBEWITHSHA256AND256BITAES-CBC-BC"
          type="javax.sql.DataSource"
          driverClassName="org.postgresql.Driver"
          url="jdbc:postgresql://127.0.0.1:5432/clover410m1?charSet=UTF-8"
          username="conf#Ws9IuHKo9h7hMjPllr31VxdI1A9LKIaYfGEUmLet9rA="
          password="conf#Cj1v59Z5nCBHaktn6Ubgst4Iz69JLQ/q6/32Xwr/IEE="
          maxActive="20" maxIdle="10"
          maxWait="-1"/>
```

Encrypted JNDI on Jetty 9 (9.2.6)

http://eclipse.org/jetty/documentation/current/configuring-security-secure-passwords.html

Configuration of JNDI jdbc connection pool is stored in the plain text file, `$JETTY_HOME/etc/jetty.xml`.

```
<New id="MysqlDB" class="org.eclipse.jetty.plus.jndi.Resource">
 <Arg></Arg>
 <Arg>jdbc/MysqlDS</Arg>
 <Arg>
  <New class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource">
   <Set name="URL">jdbc:mysql://localhost:3306/clover_empty</Set>
   <Set name="User">user</Set>
   <Set name="Password">password</Set>
  </New>
 </Arg>
</New>
```

Obfuscating password

Password can be obfuscated by using class `org.eclipse.jetty.util.security.Password` within `lib/jetty-util-{VERSION}.jar`:

```
java -cp lib/jetty-util-9.2.6.v20141205.jar org.eclipse.jetty.util.security.Password password
```

Command returns obfuscated and hashed password. The obfuscated one will be used to replace the plain password value.

Replacing password

Replace the plain password with the Call element. It's only argument is a string started with the OBF: prefix returned by the command mentioned in the previous section.

```
<New id="MysqlDB" class="org.eclipse.jetty.plus.jndi.Resource">
 <Arg></Arg>
 <Arg>jdbc/MysqlDS</Arg>
 <Arg>
  <New class="com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource">
   <Set name="URL">jdbc:mysql://localhost:3306/clover_empty</Set>
   <Set name="User">user</Set>
   <Set name="Password">
    <Call class="org.eclipse.jetty.util.security.Password" name="deobfuscate">
     <Arg>OBF:1v2j1uum1xtv1zej1zer1xtn1uvk1v1v</Arg>
    </Call>
   </Set>
  </New>
 </Arg>
</New>
```

Password in the JMS connection can be also obfuscated.

Encrypted JNDI on JBoss 6.0.0

Original datasource with unencrypted password:

```
<datasources>
 <local-tx-datasource>
  <jndi-name>MysqlDS</jndi-name>
  <connection-url>jdbc:mysql://127.0.0.1:3306/clover</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>user</user-name>
  <password>password</password>
 </local-tx-datasource>
</datasources>
```

Encrypt the data source password

Linux

```
java -cp client/jboss-logging.jar:lib/jbosssx.jar org.jboss.resource.security.SecureIdentityLoginModule password
```

Windows

```
java -cp client\jboss-logging.jar;lib\jbosssx.jar org.jboss.resource.security.SecureIdentityLoginModule password
```

NOTE: in the JBoss documentation `client/jboss-logging-spi.jar` is used, but there is no such a file in my JBossAS [6.0.0.Final "Neo"], but `client/jboss-logging.jar` can be used instead.

The command will return an encrypted password, e.g. 5dfc52b51bd35553df8592078de921bc.

Create a new application authentication policy in `conf/login-config.xml` within currently used server's profile directory (e.g. `server/default/conf/login-config.xml`).

```
<application-policy name="EncryptDBPassword">
 <authentication>
  <login-module code="org.jboss.resource.security.SecureIdentityLoginModule" flag="required">
   <module-option name="username">user</module-option>
   <module-option name="password">5dfc52b51bd35553df8592078de921bc</module-option>
   <module-option name="managedConnectionFactoryName">jboss.jca:name=MysqlDS,service=LocalTxCM</module-option>
  </login-module>
 </authentication>
```

```
</application-policy>
```

Replace authentication entries with a reference to the application authentication policy

```
<security-domain>EncryptDBPassword</security-domain>
```

Final datasource looks like this:

```
<datasources>
 <local-tx-datasource>
  <jndi-name>MysqlDS</jndi-name>
  <connection-url>jdbc:mysql://127.0.0.1:3306/clover</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <security-domain>EncryptDBPassword</security-domain>
 </local-tx-datasource>
</datasources>
```

The same mechanism can be probably used also for JMS.

```
<tx-connection-factory>
...
<security-domain-and-application>RealmWithEncryptedPassword</security-domain-and-application>
...
</tx-connection-factory>
```

See http://docs.jboss.org/jbosssecurity/docs/6.0/security_guide/html/Encrypting_Data_Source_Passwords.html

JBoss 7 (JBoss EAP 6.2.0.GA (AS 7.3.0.Final-redhat-14))

Configuration steps are similar to configuring of JBoss 6.

All configuration takes place in the single configuration file, e.g. for standalone profile `JBOSS_HOME/standalone/configuration/standalone.xml`.

Original datasource:

```
<datasources>
 <datasource jndi-name="java:/MysqlDS" pool-name="MySQLPool">
  <connection-url>jdbc:mysql://localhost:3306/clover</connection-url>
  <driver>mysql</driver>
  <pool>
   <max-pool-size>30</max-pool-size>
  </pool>
  <security>
   <user-name>user</user-name>
   <password>password</password>
  </security>
 </datasource>

 <drivers>
  <driver name="mysql" module="com.cloveretl.jdbc">
   <driver-class>com.mysql.jdbc.Driver</driver-class>
  </driver>
 </drivers>
<datasources>
```

In JBOSS_HOME directory run cli command:

```
java -cp modules/system/layers/base/org/picketbox/main/picketbox-4.0.19.SP2-redhat-1.jar:client/jboss-logging.jar
```

The command will return an encrypted password, e.g. 5dfc52b51bd35553df8592078de921bc.

Add a new security-domain to security-domains, password value is result of the command from the previous step.

```
<security-domain name="EncryptDBPassword" cache-type="default">
 <authentication>
  <login-module code="org.picketbox.datasource.security.SecureIdentityLoginModule" flag="required">
   <module-option name="username" value="user"/>
   <module-option name="password" value="5dfc52b51bd35553df8592078de921bc"/>
   <module-option name="managedConnectionFactoryName" value="jboss.jca:service=LocalTxCM,name=MysqlPool"/>
  </login-module>
 </authentication>
</security-domain>
```

Replace user and password with a reference to the security domain.

```
<datasources>
 <datasource jndi-name="java:/MysqlDS" pool-name="MysqlPool" enabled="true" use-java-context="true">
  <connection-url>jdbc:mysql://localhost:3306/clover</connection-url>
  <driver>mysql</driver>
  <pool>
   <max-pool-size>30</max-pool-size>
  </pool>
  <security>
   <security-domain>EncryptDBPassword</security-domain>
  </security>
 </datasource>

 <drivers>
  <driver name="mysql" module="com.cloveretl.jdbc">
   <driver-class>com.mysql.jdbc.Driver</driver-class>
  </driver>
 </drivers>
</datasources>
```

The same mechanism can be probably used also for JMS.

http://middlewaremagic.com/jboss/?p=1026

Encrypted JNDI on Glassfish 3 (3.1.2.2)

Configuration of jdbc connection pool is stored in the plain text file, $DOMAIN/config/domain.xml.

```
<jdbc-connection-pool driver-classname="com.mysql.jdbc.Driver" datasource-classname="" res-type="java.sql.Driver"
 <property name="URL" value="jdbc:mysql://localhost:3306/clover_empty"></property>
 <property name="user" value="user"></property>
 <property name="password" value="password"></property>
</jdbc-connection-pool>
```

Password is unencrypted, but can be replaced by so called password alias:

A password alias stores a password in encrypted form in the domain keystore, providing a clear-text alias name to use instead of the password. In password files and the domain configuration file, use the form ${ALIAS=alias-name} to refer to the encrypted password.

Creating a password alias

Password alias can be created in two ways. By using create-password-alias command in command-line admin-console utility or in the web Server Administration Console in the Password Aliases section (Domain->Password Aliases).

Replacing password with the password alias

Replace password (the value attribute) with string ${ALIAS=password_alias_name}, where password_alias_name is the name of the alias.

```
<jdbc-connection-pool driver-classname="com.mysql.jdbc.Driver" datasource-classname="" res-type="java.sql.Driver"
 <property name="URL" value="jdbc:mysql://localhost:3306/clover_empty"></property>
 <property name="user" value="user"></property>
 <property name="password" value="${ALIAS=password_alias_name}"></property>
</jdbc-connection-pool>
```

NOTE: Glassfish's Administration Server Console mentions a lower case keyword alias, but it doesn't work for me. Changing to upper case ALIAS makes the connection pool work.

Password for a JMS connection can be replaced with an alias as well.

Encrypted JNDI on WebSphere 8.5.5.0

In WebSphere user credentials aren't saved in plain text, but as J2C authentication data.

http://www-01.ibm.com/support/docview.wss?uid=nas8N1011315

The same mechanism can be used also for JMS connection.

(Configuring an external JMS provider: https://www.ibm.com/developerworks/community/blogs/timdp/entry/using_activemq_as_a_jms_provider_in_websphere_application_server_7149?lang=en)

Encrypted JNDI on WebLogic

Password in JNDI datasource file is encrypted by default when created by admin's web console (Service/Datasource).

Example of datasource file (located in `DOMAIN/config/jdbc/` directory):

```
<?xml version='1.0' encoding='UTF-8'?>
<jdbc-data-source xmlns="http://xmlns.oracle.com/weblogic/jdbc-data-source" xmlns:sec="http://xmlns.oracle.com/web
  <name>MysqlDS</name>
  <jdbc-driver-params>
    <url>jdbc:mysql://127.0.0.1:3306/clover</url>
    <driver-name>com.mysql.jdbc.Driver</driver-name>
    <properties>
      <property>
        <name>user</name>
        <value>user</value>
      </property>
    </properties>
    <password-encrypted>{AES}zIiq6/JutK/wD4CcRPX1pOueIlKqc6uRVxAnZZcC3pI=</password-encrypted>
  </jdbc-driver-params>
  <jdbc-connection-pool-params>
    <test-table-name>SQL SELECT 1</test-table-name>
  </jdbc-connection-pool-params>
  <jdbc-data-source-params>
    <jndi-name>jdbc/MysqlDS</jndi-name>
    <global-transactions-protocol>OnePhaseCommit</global-transactions-protocol>
  </jdbc-data-source-params>
</jdbc-data-source>
```

The same mechanism is also used for encrypting password in the JMS connection.

(Configuring an external JMS provider: http://docs.oracle.com/cd/E12840_01/wls/docs103/ConsoleHelp/taskhelp/jms_modules/foreign_servers/ConfigureForeignServers.html

Chapter 9. List of Properties

Table 9.1. General configuration

key	description	default
config.file	location of CloverETL Server configuration file	[working_dir]/ cloverServer.properties
license.file	location of CloverETL Server licence file (license.dat)	
engine.config.file	location of CloverETL engine configuration properties file	properties file packed with CloverETL
sandboxes.home	This property is primarily intended to be used as placeholder in the sandbox root path specification. So the sandbox path is specified with the placeholder and it's resolved to the real path just before it's used. Sandbox path may still be specified by absolute path, but placeholder has some significant advantages: * sandbox definition may be exported/imported to another environment with different directory structure * user creating sandboxed doesn't have to care about physical location on the filesystem * each node in cluster environment may have different "sandboxes.home" value, so the directory structure doen't have to be identical The default value uses configuration property "user.data.home" which points to the home directory of the user which runs the JVM process. Directory depends on the OS. On unix-like systems it's typically /home/[username]	${user.data.home}/ CloverETL/ sandboxes
private.properties	List of server properties which are used only by CloverETL Server code. So these properties are not accessible outside of the ServerFacade. By default there are all properties which may contain password in the list. Basically it means, that their values are not visible for web GUI users. Values are replaced by single star "*". Changes in this list may cause unexpected behavior of some server API.	jdbc.password, executor.password, security.ldap.password, clover.smtp.password
engine.plugins.additional.src	This property may contain absolute path to some "source" of additional CloverETL engine plugins. These plugins are not a substitute for plugins packed in WAR. "Source" may be directory or zip file. Both directory and zip must contain subdirectory for each plugin. Changes in the directory or the ZIP file apply only when the server is restarted. For details see Chapter 25, Extensibility - CloverETL Engine Plugins (p. 180).	empty
datasource.type	Set this explicitly to JNDI if you need CloverETL Server to connect to DB using JNDI datasource. In such case, parameters "datasource.jndiName" and "jdbc.dialect" must be set properly. Possible values: JNDI | JDBC	JDBC

key	description	default
datasource.jndiName	JNDI location of DB DataSource. It is applied only if "datasource.type" is set to "JNDI".	java:comp/env/jdbc/clover_server
jdbc.driverClassName	class name for jdbc driver name	
jdbc.url	jdbc url used by CloverETL Server to store data	
jdbc.username	jdbc database user name	
jdbc.password	jdbc database user name	
jdbc.dialect	hibernate dialect to use in ORM	
quartz.driverDelegateClass	SQL dialect for quartz. Value is automatically derived from "jdbc.dialect" property value.	
sandboxes.access.check.boundaries.enabled	true | false If it is set to false, then path relative to sandbox root may point out of the sandbox. No file/folder outside of the sandbox is accessible by the relative path otherwise.	true
security.session.validity	Session validity in milliseconds. When the request of logged-in user/client is detected, validity is automatically prolonged.	14400000
security.session.exchange.limit	Interval for exchange of invalid tokens in milliseconds.	360000
security.default_domain	Domain which all new users are included in. Stored in user's record in the database. Shouldn't be changed unless the "clover" must be white-labelled.	clover
security.basic_authentication.features_list	Semi-colon separated list of features which are accessible using HTTP and which should be protected by Basic HTTP Authentication. Each feature is specified by its servlet path.	/request_processor;/simpleHttpApi;/launch;/launchIt;/downloadStorage;/downloadFile;/uploadSandboxFile;/downloadLog;/webdav
security.basic_authentication.realm	Realm string for HTTP Basic Authentication.	CloverETL Server
security.digest_authentication.features_list	Semi-colon separated list of features which are accessible using HTTP and which should be protected by HTTP Digest Authentication. Each feature is specified by its servlet path. Please keep in mind, that HTTP Digest Authentication is feature added to the version 3.1. If you upgraded your older CloverETL Server distribution, users created before the upgrade cannot use the HTTP Digest Authentication until they reset their passwords. So when they reset their passwords (or the admin does it for them), they can use Digest Authentication as well as new users.	
security.digest_authentication.storeA1.enabled	Switch whether the A1 Digest for HTTP Digest Authentication should be generated and stored or not. Since there is no CloverETL Server API using the HTTP Digest Authentication by default, it's recommended to keep it disabled. Please not it's automatically enabled when any feature is	false

key	description	default
	specified in the property security.digest_authentication.features_list	
security.digest_authentication.realm	Realm string for HTTP Digest Authentication. If it is changed, all users have to reset their passwords, otherwise they won't be able to access to the server features protected by HTTP digest Authentication.	CloverETL Server
security.digest_authentication.nonce_validity	Interval of validity for HTTP Digest Authentication specified in seconds. When the interval passes, server requires new authentication from the client. Most of the HTTP clients do it automatically.	300
clover.event.fileCheckMinInterval	Interval of file checkes (in milliseconds) See File event listeners (p. 152) for details.	1000
clover.smtp.transport.protocol	SMTP server protocol. Possible values are "smtp" or "smtps".	smtp
clover.smtp.host	SMTP server hostname or IP address	
clover.smtp.port	SMTP server port	
clover.smtp.authentication	true/false If it is false, username and password are ignored	
clover.smtp.username	SMTP server username	
clover.smtp.password	SMTP server password	
clover.smtp.additional.*	Properties with prefix "clover.smtp.additional." are automatically added (without the prefix) to the Properties instance passed to the Mailer. May be useful for some protocol specific parameters. Prefix is removed.	
logging.project_name	used in log messages where it is neccessary to name the product name	CloverETL
logging.default_subdir	name of default subdirectory for all server logs; it is relative to the path specified by system property "java.io.tmpdir". Don't specify absolute path, use properties which are intended for absolute path.	cloverlogs
logging.logger.server_audit.enabled	Enable logging of operations called on ServerFacade and JDBC proxy interfaces. The name of output file is "server-audit.log" stored in the same directory as others CloverETL Server log files by default. Default logging level is DEBUG so it logs all operations which may process any change.	false
launch.log.dir	Location, where server should store launch requests logs. See Launch Services (p. 169) for details.	${java.io.tmpdir}/[logging.default_subdir]/launch where ${java.io.tmpdir} is system property
graph.logs_path	Location, where server should store Graph run logs. See Chapter 11, Logging (p. 74) for details.	${java.io.tmpdir}/[logging.default_subdir]/graph where ${java.io.tmpdir} is system property

key	description	default
temp.default_subdir	Name of default subdirectory for server tmp files; it is relative to the path specified by system property "java.io.tmpdir".	clovertmp
graph.debug_path	Location, where server should store Graph debug info.	${java.io.tmpdir}/[temp.default_subdir]/debug where ${java.io.tmpdir} is system property
graph.pass_event_params_to_graph_in_old_style	Since 3.0. It is switch for backwards compatibility of passing parameters to the graph executed by graph event. In version prior to 3.0 all params has been passed to executed graph. Since 3.0 just specified parameters are passed. Please see Task - Execution of Graph (p. 125) for details.	false
threadManager.pool.corePoolSize	Number of threads which are always active (running or idling). Related to thread pool for processing server events.	4
threadManager.pool.queueCapacity	Max size of the queue (FIFO) which contains tasks waiting for available thread. Related to thread pool for processing server events. For queueCapacity=0, there are no waiting tasks, each task is immediatelly executed in available thread or in new thread.	0
threadManager.pool.maxPoolSize	Max number of active threads. If no thread from core pool is available, pool creates new threads up to "maxPoolSize" threads. If there are more concurrent tasks then maxPoolSize, thread manager refuses to execute it.	8192
threadManager.pool.allowCoreThreadTimeOut	Switch for iddling threads timeout. If true, the "corePoolSize" is ignored so all iddling threads may be time-outed	false
threadManager.pool.keepAliveSeconds	timeout for iddling threads in seconds	20
task.archivator.batch_size	Max number of records deleted in one batch. It is used for deleting of archived run records.	50
launch.http_header_prefix	Prefix of HTTP headers added by launch services to the HTTP response.	X-cloveretl
task.archivator.archive_file_prefix	Prefix of archive files created by archivator.	cloverArchive_
license.context_names	Comma separated list of web-app contexts which may contain license. Each of them has to start with slash! Works only on Apache Tomcat.	/clover-license,/clover_license
license.display_header	Switch which specifies whether display license header in server web GUI or not.	false

Table 9.2. Defaults for job execution configuration - see Job config properties (p. 101) for details

key	description	default
executor.tracking_interval	Interval in milliseconds for scanning current status of running graph. The shorter interval, the bigger log file.	2000
executor.log_level	Log level of graph runs. TRACE | DEBUG | INFO | WARN | ERROR	INFO

key	description	default
executor.max_job_tree_depth	Defines maximal depth of the job execution tree, e.g. for recursive job it defines maximal level of recursion (counting from root job).	32
executor.max_running_concurrently	Amount of graph instances which may exist (or run) concurrently. 0 means no limits	0
executor.max_graph_instance_age	Interval in milliseconds. Specifies how long graph instance can be idling before it is released from memory. 0 means no limits. This property has been renamed since 2.8. Original name was executor.maxGraphInstanceAge	0
executor.classpath	Classpath for transformation/processor classes used in the graph. Directory [sandbox_root]/trans/ does not have to be listed here, since it is automatically added to graph run classpath.	
executor.skip_check_config	Disables check of graph configuration. Increases performance of graph execution, however may be useful during graph development.	true
executor.password	This property is deprecated. Password for decoding of encoded DB connection passwords.	
executor.verbose_mode	If true, more descriptive logs of graph runs are generated.	true
executor.use_jmx	If true, graph executor registers jmx mBean of running graph.	true
executor.debug_mode	If true, edges with enabled debug store data into files in debug directory. See property graph.debug_path (p. 68)	false

See Chapter 27, Clustering features (p. 183) for more properties.

Chapter 10. Secure configuration properties

Some configuration properties can be confidential (e.g. password to database, mail client) and thus it's desirable to encrypt them. For this purpose there is a command-line utility *secure-cfg-tool.jar*.

Basic utility usage

1. Get utility archive file (`secure-cfg-tool.zip`) and unzip it.

   The utility is available in the download section of your CloverETL account - at the same location as the download of **CloverETL Server**.

2. Execute script given for your operating system, `encrypt.bat` for MS Windows, `encrypt.sh` for Linux. You will be asked for inserting a value of configuration property intended to be encrypted.

   Example:

```
C:\secure-cfg-tool>encrypt.bat

************************************************************
Secure config encryption (use --help or -h option to show help)
************************************************************

****** Config settings ******
Provider:  SunJCE
Algorithm:  PBEWithMD5AndDES
****************************

Enter text to encrypt: mypassword
Text to encrypt: "mypassword"
Encrypted text: conf#eCflGDlDtKSJjh9VyDlRh7IftAbI/vsH

C:\secure-cfg-tool>
```

   If you want configure the way how are values encrypted, see Advanced usage (custom settings) (p. 71)

3. Encrypted string has format *conf#encrypted_property*. The encrypted string can be used as a value of configuration property in the properties file, `clover.xml` file or `web.xml` file (see details about configuration sources in Chapter 6, Config Sources and Their Priorities (p. 41)).

   Example (snippet of configuration property file):

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://example.com:3306/clover?useUnicode=true&characterEncoding=utf8
jdbc.username=example
jdbc.password=conf#eCflGDlDtKSJjh9VyDlRh7IftAbI/vsH
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

Note

Alternatively, command **java -jar secure-cfg-tool.jar** can be used.

Important

Values encrypted by Secure parameter form (Chapter 13, Secure parameters(p. 83**) cannot** be used as a value of configuration property.

Advanced usage (custom settings)

The way how configuration values are encrypted described so far, uses default configuration settings (a default provider and algorithm). But if there is a need to change these default settings with the custom ones, the *secure-cfg-tool.jar* utility offers a set of parameters to achieve that.

Table 10.1. Parameters

Parameter	Description	Example
--algorithm, -a	algorithm to encrypt	--algorithm PBEWithMD5AndDES
--file, -f	config file location	-f C:\User\John\cloverServer.properties
--help, -h	show help	--help
--providerclass, -c	custom provider class	-c org.provider.ProviderClass
--providerlocation, -l	path to jar/folder containing custom provider class (it will be added to classpath)	--providerlocation C:\User\John\lib \customprovider.jar, -l C:\User\John\lib\
--providers, -p	print available security providers and their algorithms	--providers

Note

To demonstrate usage of an external provider the Bouncy Castle provider is used.

To find out a list of algorithms use **-p** or **--providers**

```
C:\secure-cfg-tool>encrypt.bat -p
```

If you want to find out a list of algorithms of an external provider, you must pass the provider's class name and path to jar file(s)

```
C:\secure-cfg-tool>encrypt.bat -p -c org.bouncycastle.jce.provider.BouncyCastleProvider -l C:\User\John\bcprov-j
```

Result might look like this

```
***** List of available providers and their algorithms *****
Provider: SunJCE
Provider class: com.sun.crypto.provider.SunJCE
 Algorithms:
  PBEWithMD5AndDES
  PBEWithSHA1AndDESede
  PBEWithSHA1AndRC2_40
Provider: BC
Provider class: org.bouncycastle.jce.provider.BouncyCastleProvider
 Algorithms:
  PBEWITHMD2ANDDES
  PBEWITHMD5AND128BITAES-CBC-OPENSSL
  PBEWITHMD5AND192BITAES-CBC-OPENSSL
  PBEWITHMD5AND256BITAES-CBC-OPENSSL
```

Provider class is displayed on the row starting with *Provider class*, algorithms are strings with *PBE* prefix. Both can be used to configure encryption.

Configuring the encryption process

Algorithm and provider can be passed to the utility in two ways.

Using command line arguments

To change the algorithm use argument **-a**. Provider remains the default one (SunJCE in case of Oracle Java).

```
C:\secure-cfg-tool>encrypt.bat -a PBEWithMD5AndDES
```

Using of an external provider is a little more complex. Provider's class name must be specified (argument **--providerclass** or **-c**) and jar(s) must be added to the classpath (argument --providerlocation, -l). Provider location must point to concrete jar file or directory containing jar(s) and can be used several times for several paths.

```
C:\secure-cfg-tool>encrypt.bat -a PBEWITHSHA256AND256BITAES-CBC-BC -c org.bouncycastle.jce.provider.BouncyCastle
```

Using configuration file

Configuration file is common properties file (text file with key-value pairs):

```
[property-key]=[property-value]
```

It might look like this (example comes from `secure.config.example.properties`, distributed within `secure-cfg-tool.zip`):

```
security.config_properties.encryptor.providerClassName=org.bouncycastle.jce.provider.BouncyCastleProvider
security.config_properties.encryptor.algorithm=PBEWITHSHA256AND256BITAES-CBC-BC
security.config_properties.encryptor.provider.location=C:\\User\\libs
```

To let utility know about the configuration file use **-f** argument

```
C:\secure-cfg-tool>encrypt.bat -f secure.config.example.properties
```

Note

More jar locations can be set in the **security.config_properties.encryptor.providerLocation**, locations are delimited by semicolon.

Configuring an application server

CloverETL Server application needs to know how the values have been encrypted, therefore the properties must be passed to the server (see details in Part III, “Configuration” (p. 40)). For example:

```
...
security.config_properties.encryptor.providerClassName=org.bouncycastle.jce.provider.BouncyCastleProvider
security.config_properties.encryptor.algorithm=PBEWITHSHA256AND256BITAES-CBC-BC
...
```

Important

If a third-party provider is used, its classes must be accessible for the application server. Property **security.config_properties.encryptor.providerLocation** will be ignored.

Chapter 11. Logging

Main logs

The CloverETL Server uses the log4j library for logging. The WAR file contains the default log4j configuration. The log4j configuration file `log4j.xml` is placed in `WEB-INF/classes` directory.

By default, log files are produced in the directory specified by system property "java.io.tmpdir" in the `cloverlogs` subdirectory.

"java.io.tmpdir" usually contains common system temp dir i.e. `/tmp`. On Tomcat, it is usually `$TOMCAT_HOME/temp`

The default logging configuration (log4j.xml bundled in the clover.war) may be changed to another log4j configuration file using system property `log4j.configuration`. If you override the configuration, only the properties from the new file are used.

The `log4j.configuration` should contain the URL of the new log4j configuration file. It's not just file system path, it must be URL, so it may look like this:

```
log4j.configuration=file:/home/clover/config/log4j.xml
```

It is better to copy the original file and modify the copy, than to create a new one.

Please note, that "log4j.configuration" is not a CloverETL Server config property, but system property, thus it must be set on the JVM command line by -Dlog4j.configuration or in other way suitable for the application container. Best possibility how to set system property for each application container is described in the "Installation" chapter.

Since such a configuration overrides the default configuration, it may have influence over Graph run logs. So your own log config has to contain following fragment to preserve Graph run logs

```
<logger name="Tracking" additivity="false">
  <level value="debug"/>
</logger>
```

Another useful logging settings

These system properties allow for logging of HTTP requests/responses to stdout:

Client side:

`com.sun.xml.ws.transport.http.client.HttpTransportPipe.dump=true` (for more information consult CloverETL Designer Users's Guide - chapter Integrating CloverETL Designer with CloverETL Server)

Server side:

`com.sun.xml.ws.transport.http.HttpAdapter.dump=true`

Access Log in Apache Tomcat

If you need to log all requests processed by server, add the following code to `$CATALINA_HOME/conf/server.xml`.

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
```

```
        prefix="localhost_access_log." suffix=".txt"
        pattern="%h %l %u %t %D %r %s %b" />
```

The format defined above has following meaning

```
[IP address] [date-time] [processing duration in millis] [method] [URL] [protocol+versi
```

The log will look like the next line

```
172.17.30.243 - - [13/Nov/2014:12:53:03 +0000] 2 "POST /clover/sDispatcher/clusterNodeA
```

Graph run logs

Each graph or jobflow run has its own log file – for example, in the Server Console, section "Executions History".

By default, these log files are saved in the subdirectory cloverLogs/graph in the directory specified by "java.io.tmpdir" system property.

It’s possible to specify a different location for these logs by the CloverETL property "graph.logs_path". This property does not influence main Server logs.

Server Audit logs

It loggs operations called on ServerFacade and JDBC proxy interfaces.

By default, this logging is disabled and could be enabled by setting the value of CloverETL property "logging.logger.server_audit.enabled" to true.

The name of output file is `server-audit.log`, in the same directory as main server log files. Default log level is DEBUG, so all operations which may do any change or another important operations (e.g. login or openJdbcConnection) are logged. To enable logging of all operations, change log level to TRACE in the log4j configuration.

Each logged operation is logged by two messages: entering method and exitting method (if the exception is raised, it's logged instead of output parameters)

- Entering method (marked as "inputParams"). All method's parameters (except for passwords) are printed.
- Exiting method (marked as "outputParams"). Method's return value is printed.
- Exception in method (marked as "EXCEPTION"). Exception's stacktrace is printed.

Message also contains:

- username, if the user is known
- client IP address, if it's known
- cluster node ID
- Interface name and the operation name

Values of transient and lazy initialized (in entity classes) fields and fields with binary content are not printed.

Part IV. Administration

Chapter 12. Temp Space Management

Many of the components available in the CloverETL Server require temporary files or directories in order to work correctly. *Temp space* is a physical location on the file system where these files or directories are created and maintained. CloverETL Server allows you to configure and manage temp spaces - you can specify their locations, see usage of the filesystem etc.

Overview

The overview of temp spaces defined in CloverETL Server is available under *Configuration* > *Temp space management* > *Overview*

The overview panel displays list of temp spaces for each node in the cluster. These properties are displayed for each temp space:

- **Root Path** - location of the temp space with unresolved placeholders (see note below for placeholders)
- **Resolved Path** - location of the temp space with resolved placeholders (see note below for placeholders)
- **Free Space** - remaining space for the temp space
- **Filesystem Size** - all available space for the temp space (actual size of the filesystem where the temp space resides)
- **Filesystem Usage** - size of used space in percentage
- **Available** - the directory exists and is writable
- **Status** - current status of temp space, can be `Active` or `Suspended`

Note

It is possible to use system properties and environment variables as placeholders. See Using environment variables and system properties (p. 79).

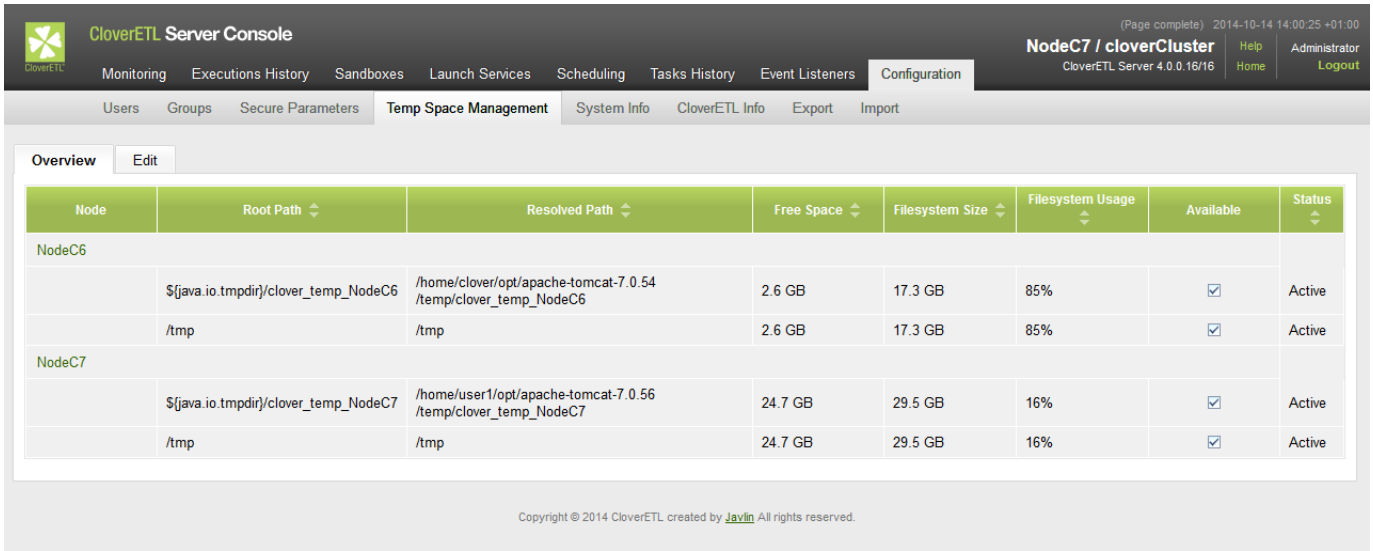


Figure 12.1. Configured temp spaces overview - one default temp space on each cluster node

Management

Temp space management offers an interface to add, disable, enable and delete a temp space. It is accessible under *Configuration > Temp space management > Edit.*

The screen is divided in two drop-down areas: Global Configuration and Per Node Configuration. The *Global configuration* manages temp spaces of standalone server or in case of a server cluster temp spaces on all its nodes. The *Per Node Configuration* allows to maintain temp spaces on each particular node.

Initialization

When CloverETL Server is starting the system checks temp space configuration: in case no temp space is configured a new default temp space is created in the directory where `java.io.tmpdir` system property points. The directory is named as follows:

- `${java.io.tmpdir}/clover_temp` in case of a standalone server
- `${java.io.tmpdir}/clover_temp_<node_id>` in case of server cluster

Adding Temp Space

In order to define new temp space enter its path into text field under last row in the table and click the **Add** link. If the directory entered does not exist, it will be created.

Tip

The main point of adding additional temp spaces is to enable higher system throughput - therefore the paths entered should point to directories residing on different physical devices to achieve maximal I/O performance.

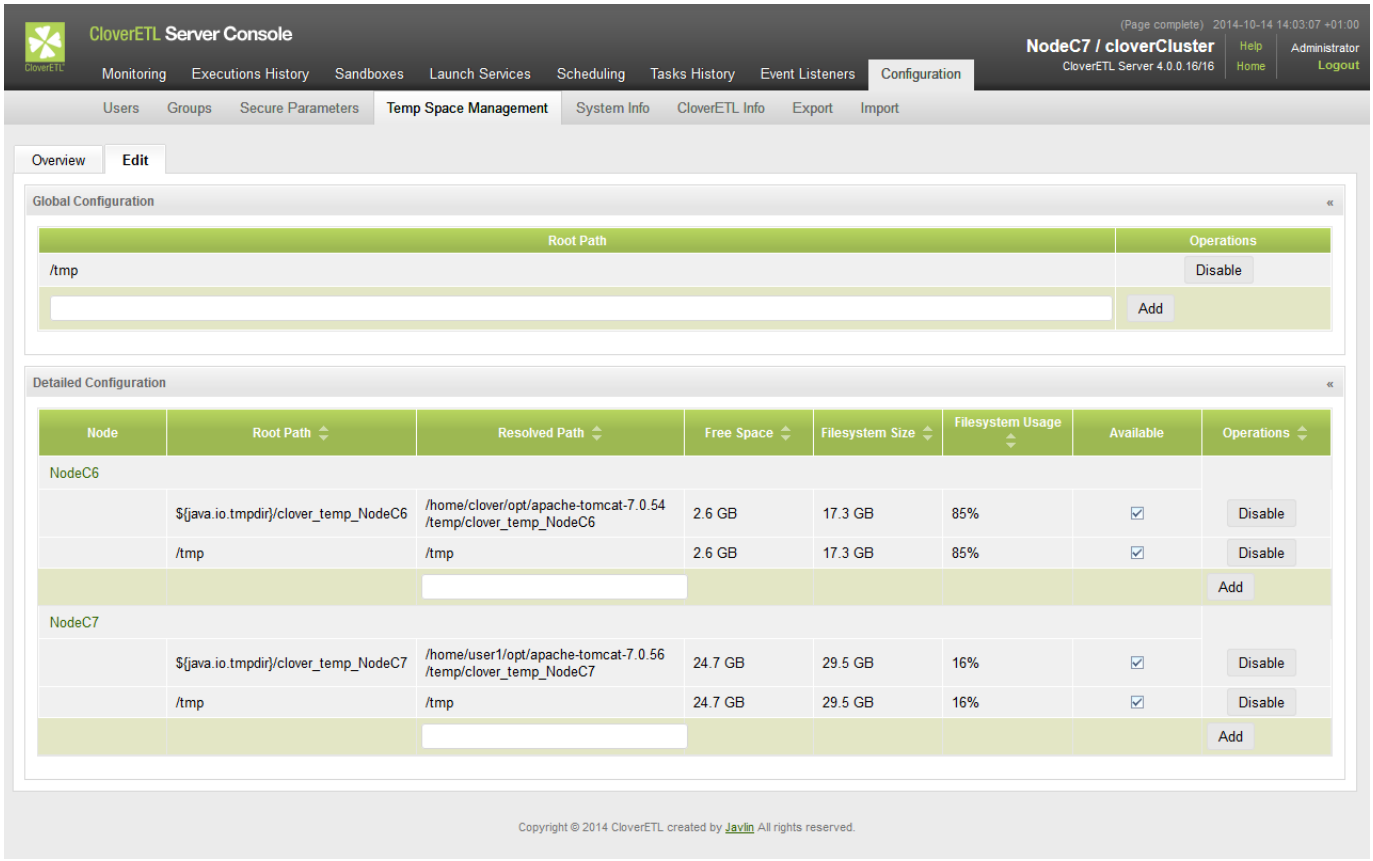


Figure 12.2. Newly added global temp space.

Using environment variables and system properties

Environment variables and system properties can be used in the temp space path as a placeholder; they can be arbitrarily combined and resolved paths for each node may differ in accord with its configuration.

Note

The environment variables have higher priority than system properties of the same name. The path with variables are resolved after system has added new temp space and when the server is starting. In case the variable value has been changed it is necessary to restart the server so that the change takes effect.

Examples:

- Given that an environment variable `USERNAME` has a value `Filip`. and is used as a placeholder in the path `C:\Users\${USERNAME}\tmp`, the resolved path is `C:\Users\Filip\tmp`.
- Given that Java system property `java.io.tmpdir` has a value `C:\Users\Filip\AppData\Local\Temp` and the property is used as a placeholder in the path `${java.io.tmpdir}\temp_folder`, the resolved path is `C:\Users\Filip\AppData\Local\Temp\temp_folder`.
- Node `node01` has been started with parameter `-Dcustom.temporary.dir=C:\tmp_node01` and node `node02` has been started with parameter `-Dcustom.temporary.dir=C:\tmp_node02`, the declared path is `${custom.temporary.dir}`. The resolved path is different for each node, `C:\tmp_node01` for `node01` and `C:\tmp_node02` for `node02`.
- When the declared path is `${java.io.tmpdir}\${USERNAME}\tmp_folder`, the resolved path is `C:\tmp\Filip\tmp_folder`.

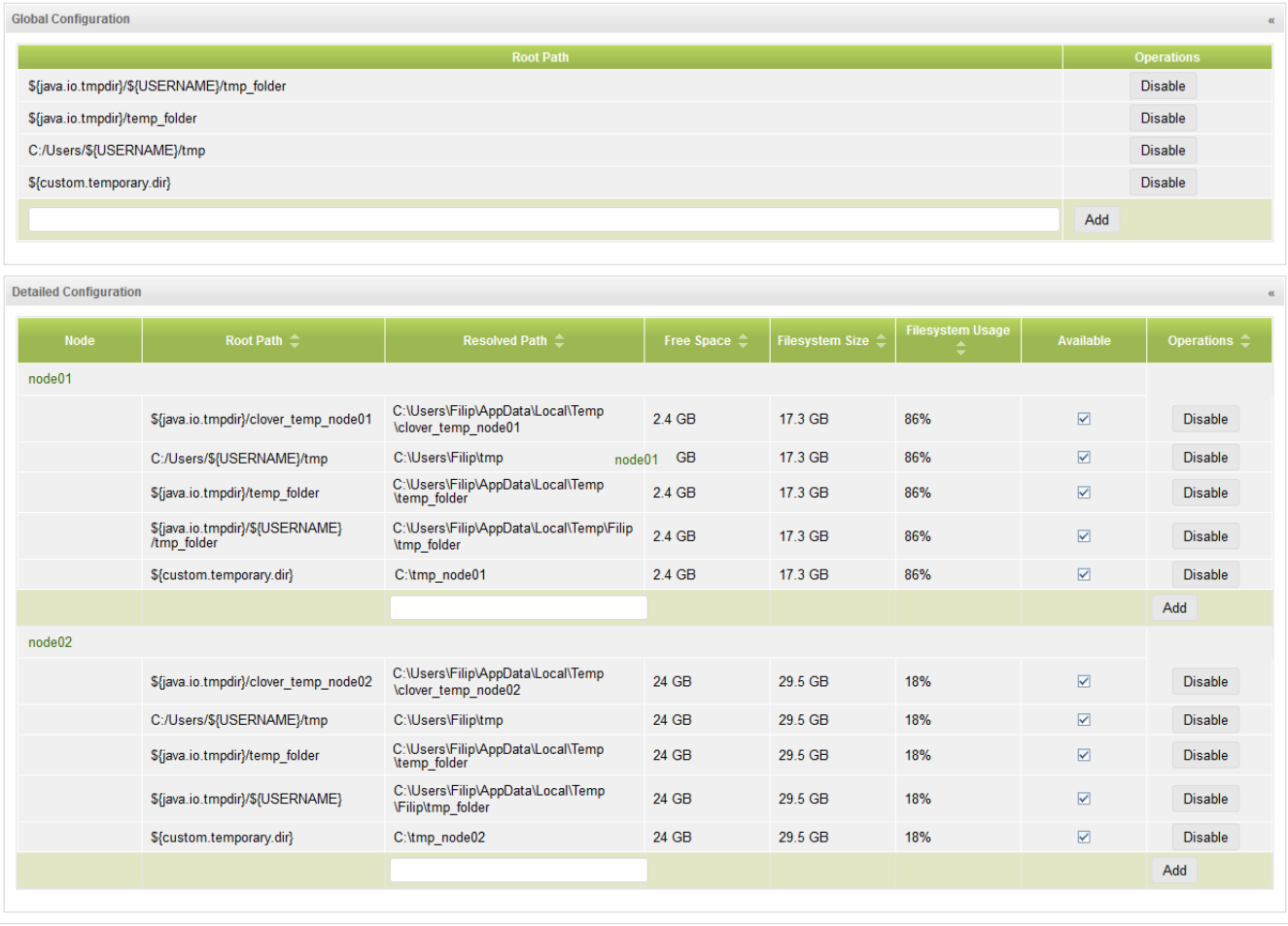


Figure 12.3. Temp spaces using environment variables and system properties

Disabling Temp Space

To disable a temp space click on "Disable" link in the panel. Once the temp space has been disabled, no new temporary files will be created in it, but the files already created may be still used by running jobs. In case there are files left from previous or current job executions a notification is displayed.

Note

The system ensures that at least one enabled temp space is available.

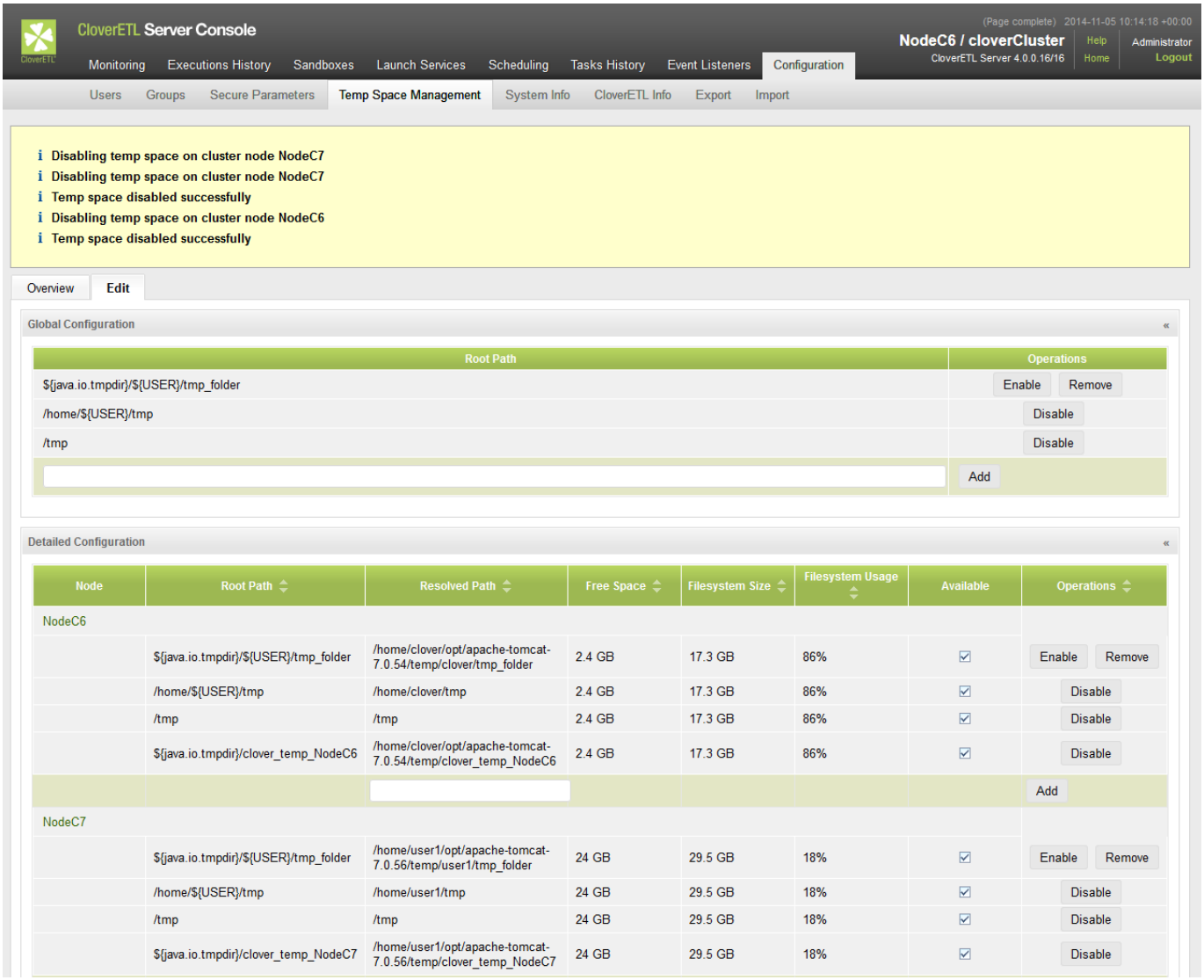


Figure 12.4. Disable operation reports action performed

Enabling Temp Space

To enable a temp space click on "Enable" link in the panel. Enabled temp space is active, i.e. available for temporary files and directories creation.

Removing Temp Space

To remove a temp space click on "Remove" link in the panel. Only disabled temp space may be removed. Should be there any running jobs using the temp space, system will not allow its removal. In case there are some files left in the temp space directory, it is possible to remove them in the displayed notification panel. The available options are:

- *Remove* - remove temp space from system, but keep its content
- *Remove and delete* - remove the temp space from system and its content too
- *Cancel* - do not proceed with operation

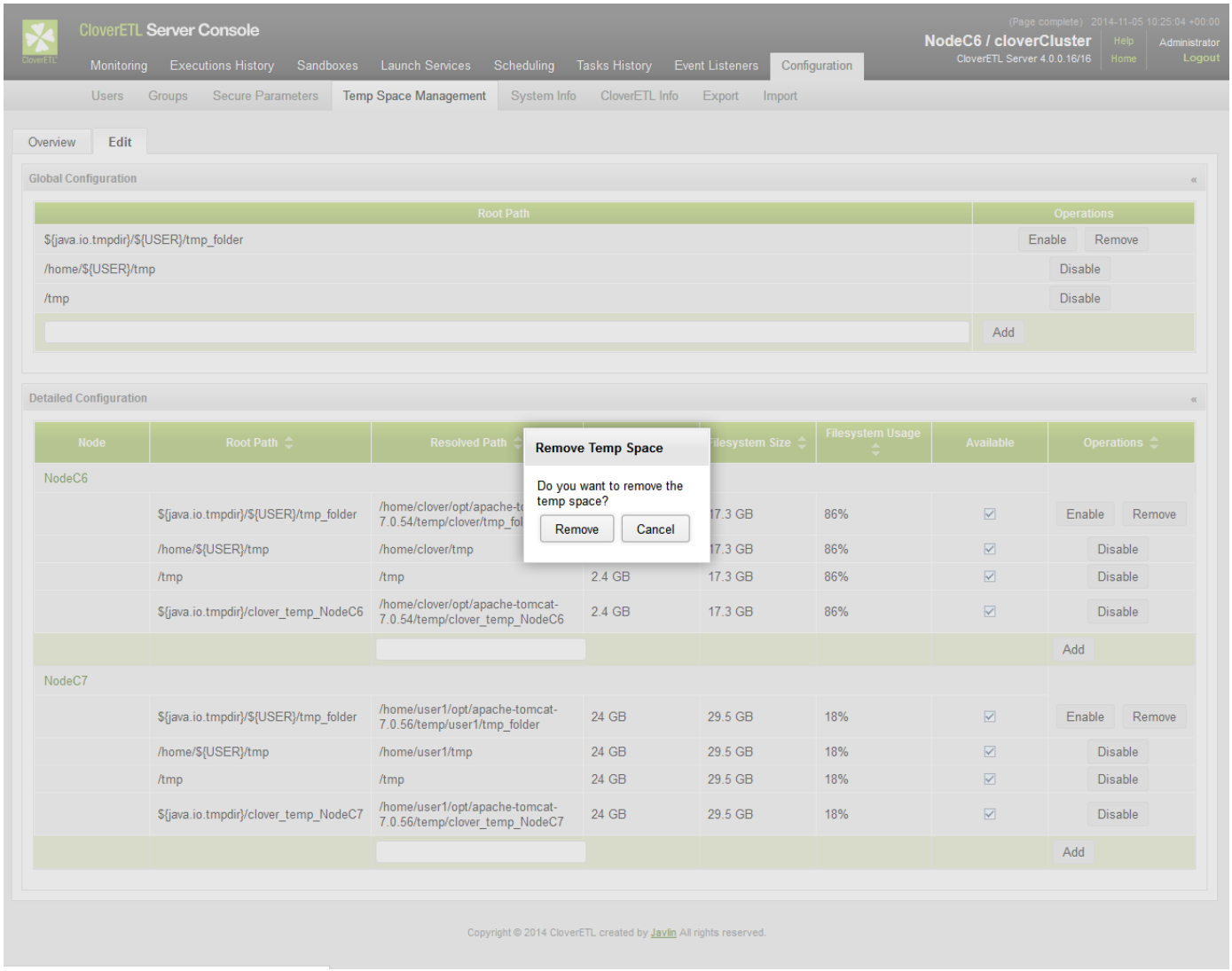


Figure 12.5. Remove operation asks for confirmation in case there are data present in the temp space

Chapter 13. Secure parameters

Transformation graphs in **CloverETL Server** environment allow you to define secure graph parameters. Secure graph parameters are regular graph parameters, either internal or external (in a *.prm file), but the values of the graph parameters are not stored in plain text on the file system - encrypted values are persisted instead. This allows you to use graph parameters to handle sensitive information, typically credentials such as passwords to databases.

Secure parameters are available only in **CloverETL Server** environment, including working with **CloverETL Server Projects** in **CloverETL Designer**.

The encryption algorithm must be initialized with a **master password**. The master password has to be manually set after server installation in *Configuration* > *Secure Parameters* > *Master password*. Secure parameters cannot be used before the master password is set.

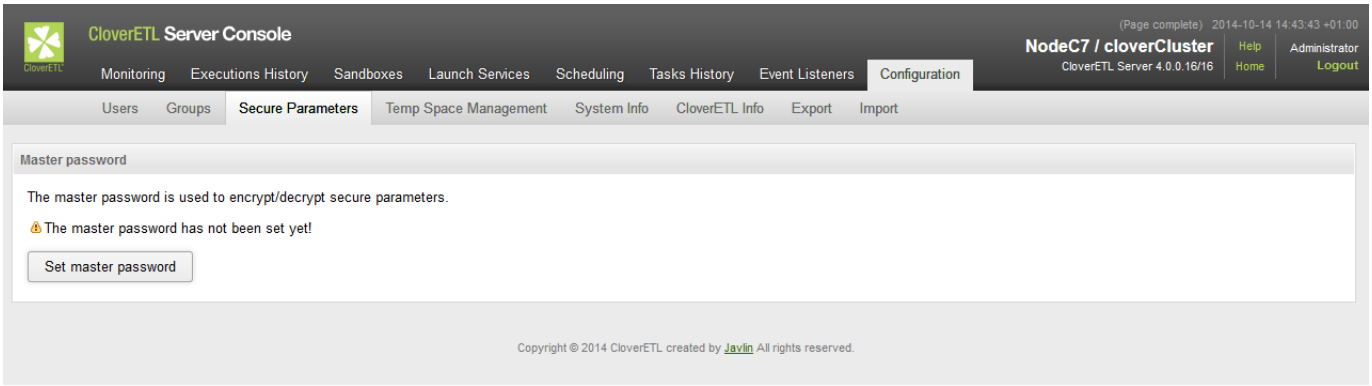


Figure 13.1. Master password initialization

After setting the master password secure parameters are fully available in **Graph parameter editor** in **CloverETL Designer**. When setting value of a secure parameter, it will be automatically encrypted using the master password. Secure parameters are automatically de-crypted by server in graph runtime. A parameter value can also be encrypted in the **CloverETL Server Console** in the *Configuration* > *Secure Parameters* page - use the **Encrypt text** section.

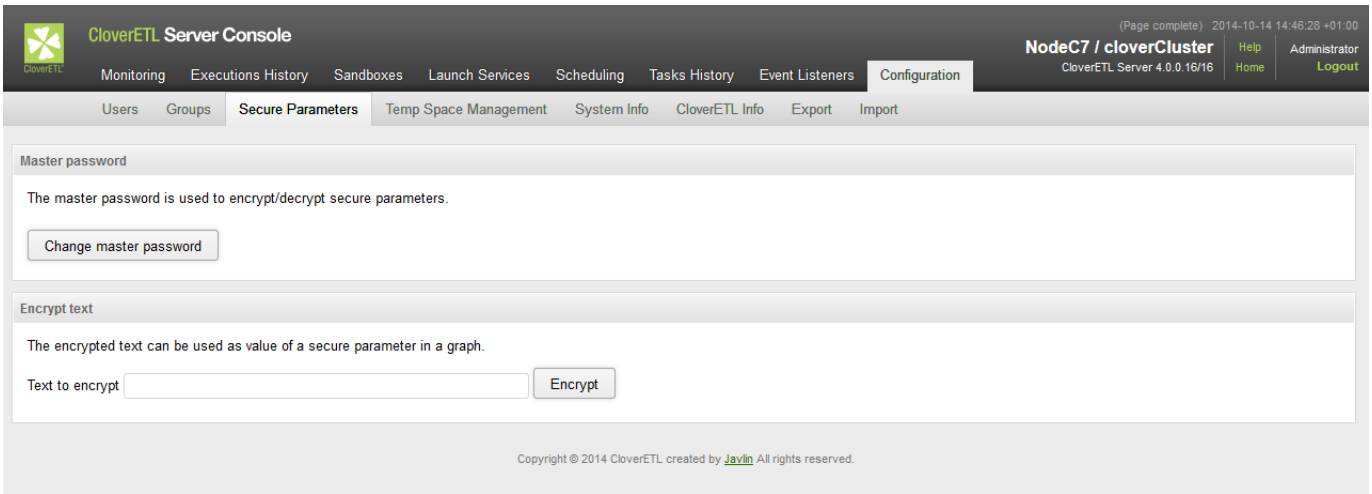


Figure 13.2. Graph parameters tab with initialized master password

If you change the master password, the secure parameters encrypted using the old master password cannot be de-crypted correctly anymore. In that case existing secure parameters need to be encrypted again with the new master password. That can be accomplished simply by setting their value (non-encrypted) again in the **Graph parameter editor**. Similar master password inconsistency issue can occur if you move a transformation graph

with some secure parameters to an another server with different master password. So it is highly recommended to use identical master password for all your **CloverETL Server** installations.

See documentation of secure parameters in **CloverETL Designer** manual for further details.

Secure parameters configuration

Encryption of secure parameters can be furter customized via server configuration parameters.

Table 13.1. Secure parameters configuration parameters

Property name	Default value	Description
security.job_parameters.encryptor.algorithm	PBEWithMD5AndDES	The algorithm to be used for encryption. This algorithm has to be supported by your JCE provider (if you specify a custom one, or the default JVM provider if you don't). The name of algorithm should start with *PBE* prefix. The list of available algorithms depends on your JCE provider, e.g. for the default *SunJCE* provider you can find them on http://docs.oracle.com/javase/6/docs/technotes/guides/security/SunProviders.html#SunJCEProvider or for the *Bouncy Castle* provider on http://www.bouncycastle.org/specifications.html (section *Algorithms/PBE*)).
security.job_parameters.encryptor.master_password_encryption	clover	The password used to encrypt values persisted in the database table *secure_param_passwd* (the master password is persisted there).
security.job_parameters.encryptor.providerClassName	Empty string. The default JVM provider is used (e.g. for Oracle Java the SunJCE provider is used)	The name of the security provider to be asked for the encryption algorithm. It must implement interface *java.security.Provider*. For example set to *org.bouncycastle.jce.provider.BouncyCastleProvider* for the *Bouncy Castle* JCE provider, see below.

Installing Bouncy Castle JCE provider

Algorithms provided by JVM could be too weak to satisfy an adequate security. Therefore it is recommended to install a third-party JCE provider. Following example demonstrates installation of one concrete provider, *Bouncy Castle* JCE provider. Another provider would be installed similarly.

1. Download Bouncy Castle provider jar (e.g. bcprov-jdk15on-150.jar) from http://bouncycastle.org/latest_releases.html

2. Add the jar to the classpath of your application container running **CloverETL Server**, e.g. to directory `WEB-INF/lib`

3. Set value of the attribute *security.job_parameters.encryptor.providerClassName* to *org.bouncycastle.jce.provider.BouncyCastleProvider* in file `WEB-INF/config.properties`.

4. Set value of the attribute *security.job_parameters.encryptor.algorithm* to the desired algorithm (e.g. *PBEWITHSHA256AND256BITAES-CBC-BC*)

Example of configuration using Bouncy Castle:

```
security.job_parameters.encryptor.algorithm=PBEWITHSHA256AND256BITAES-CBC-BC
security.job_parameters.encryptor.providerClassName=org.bouncycastle.jce.provider.BouncyCastleProvider
```

Chapter 14. Users and Groups

The CloverETL Server has a built-in security module that manages users and groups. User groups control access permissions to sandboxes and operations the users can perform on the Server, including authenticated calls to Server API functions. A single user can belong to multiple groups.

LDAP or Active Directory can be configured with the Server to authenticate users and optionally assign their effective groups (and permissions) from a global directory.

You can manage users and user groups in **Configuration/Users and Groups**. Please note that you need a “List users” (“List groups” respectively) permission for that.

LDAP Authentication

Since 3.2 it's possible to configure CloverETL Server to use LDAP server for users authentication. So the credentials of users registered in LDAP may be used for authentication to any CloverETL Server interface (API or web console).

However authorization (access levels to sandboxes content and privileges for operations) is still handled by Clover security module. Each user, event though logged-in using LDAP authentication, must have his own "user" record (with related groups) in CloverETL security module. So there must be the user with the same username and domain set to "LDAP". Such record has to be created by Server administrator before the user can log in.

What does the CloverETL do to authenticate an LDAP user?

1. User specifies the LDAP credentials in login form to the Server web console
2. CloverETL Server looks up user record and checks whether has "LDAP" domain set
3. If the system is configured to use LDAP for authentication only, it attempts to connect to LDAP server using user's credentials. If it succeeds, the user is logged in.
4. In case the system is configured for user group synchronization the procedure is as follows:
5. CloverETL Server connects to the LDAP server and checks whether the user exists (it uses specified search to lookup in LDAP).
6. If the user exists in LDAP, CloverETL Server performs authentication.
7. If succeeded, CloverETL Server searches LDAP for user's groups.
8. Clover user is assigned to the Clover groups according to his current assignation to the LDAP groups.
9. User is logged-in.

Note

Switching domains:

- If a user was **created as LDAP** and then switched to clover domain, you have to **set a password** for him in **Change password tab**.
- If a user was **created as clover** and then switched to LDAP domain, he has a password in clover domain, but it is overriden by the LDAP password. After switching back to clover domain, the **original password is re-used**. It can be reset in the **Change password** tab if needed (e.g. forgotten).

Configuration

By default CloverETL Server allows only its own internal mechanism for authentication. To enable authentication with LDAP, set the configuration property "security.authentication.allowed_domains" properly. It is a list of user domains that are used for authentication.

Currently there are 2 authentication mechanism implemented: "LDAP" and "clover" ("clover" is identifier of CloverETL internal authentication and may be changed by security.default_domain property, but only for white-labelling purposes). To enable LDAP authentication, set value to "LDAP" (only LDAP) or "clover,LDAP". Users from both domain may login. It's recommended to allow both mechanisms together, until the LDAP is properly configured. So the admin user can still login to web GUI although the LDAP connection isn't properly configured.

You can use **Setup** to configure LDAP authentication. See LDAP (p. 48) in Chapter 7, Setup (p. 43).

Basic LDAP connection properties

```
# Implementation of context factory
security.ldap.ctx_factory=com.sun.jndi.ldap.LdapCtxFactory
# URL of LDAP server
security.ldap.url=ldap://hostname:port
# User DN pattern that will be used to create LDAP user DN from login name.
security.ldap.user_dn_pattern=uid=${username},dc=company,dc=com
```

Depending on the LDAP server configuration the property `security.ldap.user_dn_pattern` can be pattern for user's actual distinguished name in the LDAP directory, or just the login name - in such case just set the property to `${username}`.

Configuration of user and group lookup

In order to be able to synchronize the Clover groups with those defined in LDAP directory, the `security.ldap.user_dn_pattern` has to be left unspecified. There are additional properties required so that the server is able to search the LDAP directory.

```
# User DN of a user that has sufficient privileges to search LDAP for users and groups
security.ldap.userDN=cn=Manager,dc=company,dc=com
# The password for user mentioned above.
security.ldap.password=
```

There are optional settings affecting how the LDAP directory is searched.

```
# Timeout for queries searching the directory.
security.ldap.timeout=5000
# Maximal number of records that the query can return.
security.ldap.records_limit=2000
# How LDAP referrals are processed, possible values are: 'follow', 'ignore' and 'throw'.
# The default depends on the context provider.
security.ldap.referral=
```

Specified values work for this specific LDAP tree:

- dc=company,dc=com
  - ou=groups
    - cn=admins (objectClass=groupOfNames,member=(uid=smith,dc=company,dc=com),member=(uid=jones,dc=company,dc=com))
    - cn=developers (objectClass=groupOfNames,member=(uid=smith,dc=company,dc=com))

    - cn=consultants (objectClass=groupOfNames,member=(uid=jones,dc=company,dc=com))
- ou=people
    - uid=smith (fn=John,sn=Smith,mail=smith@company.com)
    - uid=jones (fn=Bob,sn=Jones,mail=jones@company.com)

Following properties are necessary for lookup for the LDAP user by his username. (step [4] in the login process above)

```
# Base specifies the node of LDAP tree where the search starts
security.ldap.user_search.base=dc=company,dc=eu
# Filter expression for searching the user by his username.
# Note, that this search query must return just one record.
# Placeholder ${username} will be replaced by username specified by the logging user.
security.ldap.user_search.filter=(uid=${username})
# Scope specifies type of search in "base". There are three possible values: SUBTREE | ONELEVEL | OBJECT
# http://download.oracle.com/javase/6/docs/api/javax/naming/directory/SearchControls.html
security.ldap.user_search.scope=SUBTREE
```

Following properties are names of attributes from the search defined above. They are used for getting basic info about the LDAP user in case the user record has to be created/updated by Clover security module: (step [6] in the login process above)

```
security.ldap.user_search.attribute.firstname=fn
security.ldap.user_search.attribute.lastname=sn
security.ldap.user_search.attribute.email=mail
# This property is related to the following step "searching for groups".
# Groups may be obtained from specified user's attribute, or found by filter (see next paragraph)
# Leave this property empty if the user doesn't have such attribute.
security.ldap.user_search.attribute.groups=memberOf
```

In the following step, clover tries to find groups which the user is assigned to. (step [4] in the login process above). There are two ways how to get list of groups which the user is assigned to. The user-groups relation is specified on the "user" side. The user record has some attribute with list of groups. It's "memberOf" attribute usually. Or the relation is specified on the "group" side. The group record has attribute with list of assigned users. It's "member" attribute usually.

In case the relation is specifien on users side, please specify property:

```
security.ldap.user_search.attribute.groups=memberOf
```

Leave it empty otherwise.

In case the relation is specified on the groups side, set properties for searching:

```
security.ldap.groups_search.base=dc=company,dc=com
# Placeholder ${userDN} will be replaced by user DN found by the search above
# If the filter is empty, searching will be skipped.
security.ldap.groups_search.filter=(&(objectClass=groupOfNames)(member=${userDN}))
security.ldap.groups_search.scope=SUBTREE
```

Otherwise, please leave property security.ldap.groups_search.filter empty, so the search will be skipped.

Clover user record will be assigned to the clover groups according to the LDAP groups found by the search (or the attribute). (Groups synchronization is performed during each login)

```
# Value of the following attribute will be used for lookup for the Clover group by its code.
# So the user will be assigned to the Clover group with the same "code"
security.ldap.groups_search.attribute.group_code=cn
```

Web GUI section Users

This section is intended to users management. It offers features in dependence on user's permissions. i.e. User may enter this section, but cannot modify anything. Or user may modify, but cannot create new users.

All possible features of users section:

- *create new user*
- *modify basic data*
- *change password*
- *disable/enable user*
- *assign user to groups* - Assignment to groups gives user proper permissions

Table 14.1. After default installation on empty DB, admin user is created automatically

User name	Description
clover	Clover user has admin permissions, thus default password "clover" should be changed after installation.

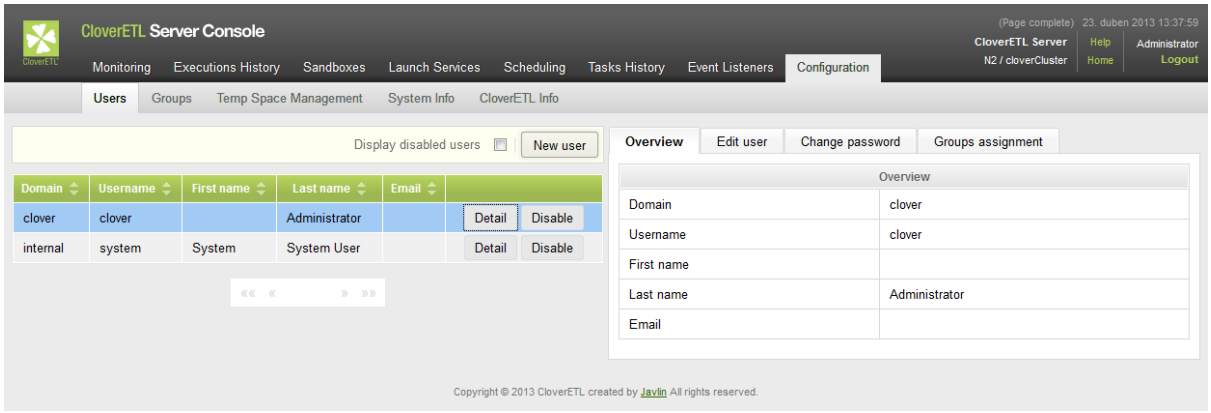


Figure 14.1. Web GUI - section "Users" under "Configuration"

Table 14.2. User attributes

Attribute	Description
Domain	Domain which is the origin of the user. There are only two possible values currently: "clover" or "ldap".
Username	Common user identifier. Must be unique, cannot contain spaces or special characters, just letters and numbers.
Password	Case sensitive password. If user looses his password, the new one must be set. Password is stored in encrypted form for security reasons, so it cannot be retrieved from database and must be changed by the user who has proper permission for such operation.
First name	
Last name	
Email	Email which may be used by CloverETL administrator or by CloverETL server for automatic notifications. See Task - Send Email (p. 139) for details.

Edit user record

User with permission "Create user" or "Edit user" can use this form to set basic user parameters.

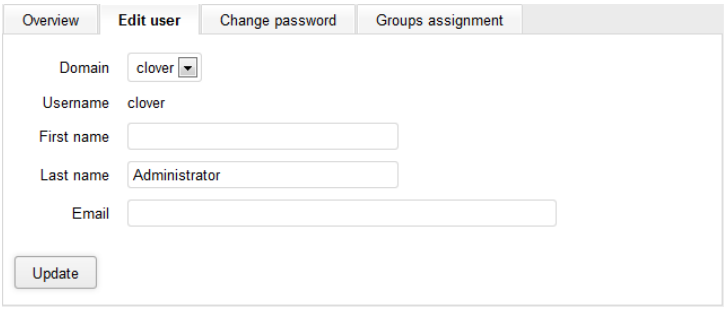


Figure 14.2. Web GUI - edit user

Change users Password

If user looses his password, the new one must be set. So user with permission "Change passwords" can use this form to do it.

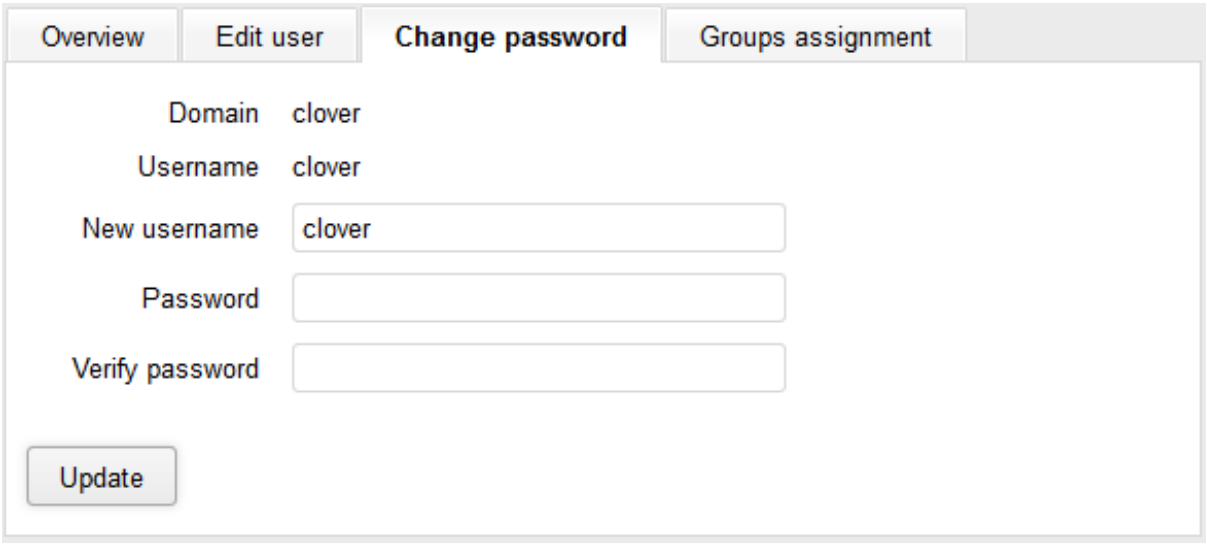


Figure 14.3. Web GUI - change password

Group assignment

Assignment to groups gives user proper permissions. Only logged user with permission "Groups assignment" can access this form and specify groups which the user is assigned in. See Web GUI section Groups (p. 93) for details about permissions.

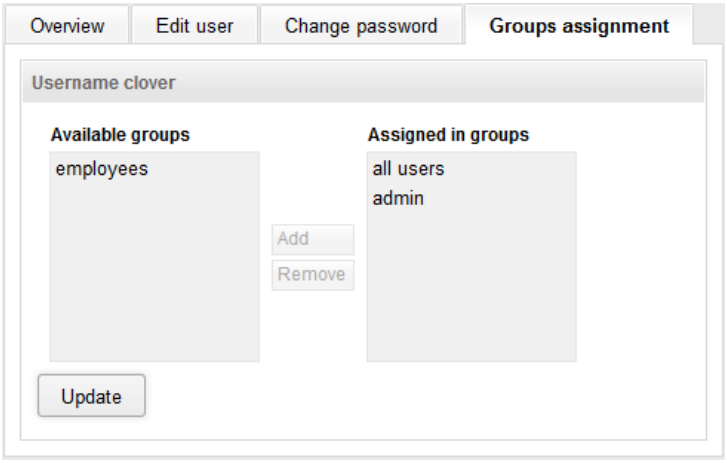


Figure 14.4. Web GUI - groups assignment

Disabling / enabling users

Since user record has various relations to the logs and history records, it can't be deleted. So it's disabled instead. It basically means, that the record doesn't display in the list and the user can't login.

However disabled user may be enabled again. Please note, that disabled user is removed from its groups, so groups should be assigned properly after re-enabling.

Web GUI section Groups

Group is abstract set of users, which gives assigned users some permissions. So it is not necessary to specify permission for each single user.

There are independent levels of permissions implemented in CloverETL Server

- *permissions to Read/Write/eXecute in sandboxes* - sandbox owner can specify different permissions for different groups. See Sandbox Content Security and Permissions (p. 96) for details.
- *permissions to perform some operation* - user with operation permission "Permission assignment" may assign specific permission to existing groups.
- *permissions to launch specific service* - see Launch Services (p. 169) for details.

Table 14.3. Default groups created during installation

Group name	Description
admins	This group has operation permission "all" assigned, which means, that it has unlimited permission. Default user "clover" is assigned to this group, which makes him administrator.
all users	Every single CloverETL user is assigned to this group by default. It is possible to remove user from this group, but it is not a recommended approach. This group is useful for some permissions to sandbox or some operation, which you would like to make accessible for all users without exceptions.

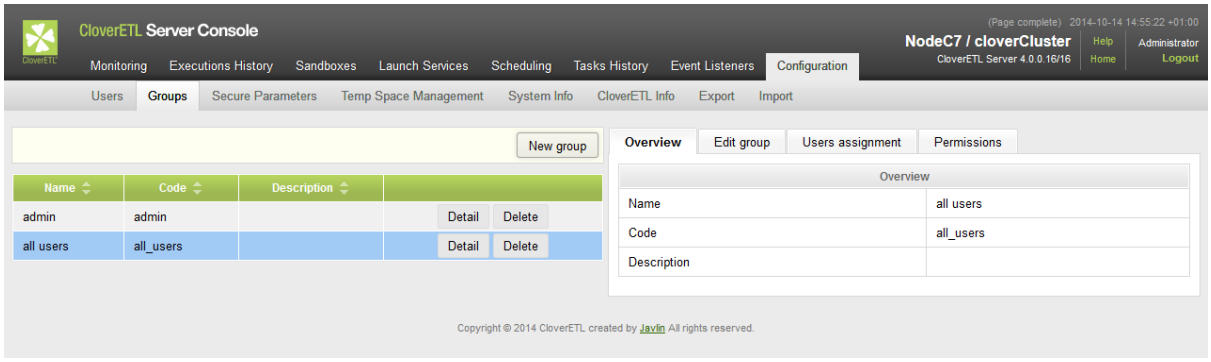


Figure 14.5. Web GUI - section "Groups"

Users Assignment

Relation between users and groups is N:M. Thus in the same way, how groups are assignable to users, users are assignable to groups.

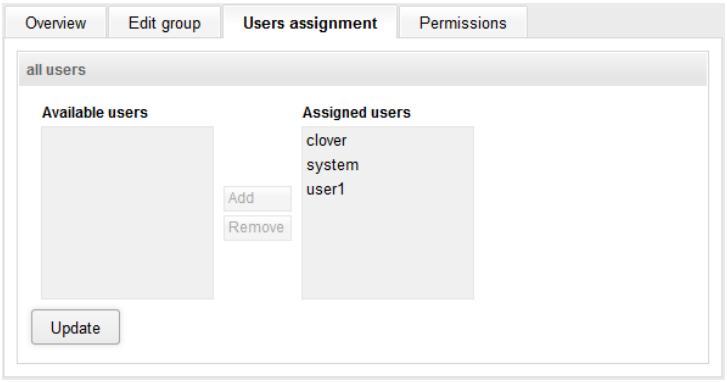


Figure 14.6. Web GUI - users assignment

Group Permissions

Groups permissions are structured as tree, where permissions are inherited from root to leafs. Thus if some permission (tree node) is enabled (blue dot), all permissions in sub tree are automatically enabled (white dot). Permissions with red cross are disabled.

Thus for "admin" group just "all" permission is assigned, every single permission in sub tree is assigned automatically.

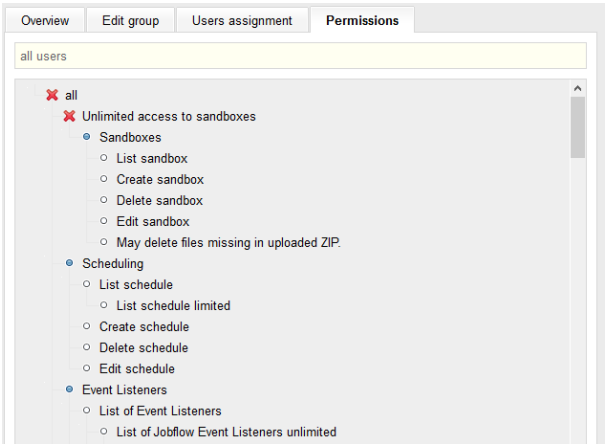


Figure 14.7. Tree of permissions

Chapter 15. Server Side Job files - Sandboxes

A sandbox is where you store all your project's transformation graph files, jobflows, data, and other resources. It's a server side analogy to a Designer project. The Server adds additional features to sandboxes, like user permissions management and global per-sandbox configuration options.

The Server and the Designer are integrated so that you are able to connect to a Server sandbox using a "Server Project" in your Designer workspace. Such a project works like a remote file system – all data is stored on the Server and accessed remotely. Nonetheless, you can do everything with Server Projects the same way as with local projects – copy and paste files, create, edit, and debug graphs, etcetera. See the **CloverETL Designer manual** for details on configuring a connection to the Server.

Technically, a sandbox is a dedicated directory on the Server host file system and its contents are managed by the Server. Advanced types of sandboxes, like "partitioned sandbox" have multiple locations to allow distributed parallel processing (more about that in Chapter 27, Clustering features (p. 183)). A sandbox cannot contain another sandbox within – it's a single root path for a project.

It's recommended to put all sandboxes in a folder outside the CloverETL Server installation (by default the sandboxes would be stored in the ${user.data.home}/CloverETL/sandboxes, where the "user.data.home" is automatically detected user home directory). However, each sandbox can be located on the file system independently of the others if needed. The containing folder and all its contents must have read/write permission for the user under which the CloverETL Server/application server is running.

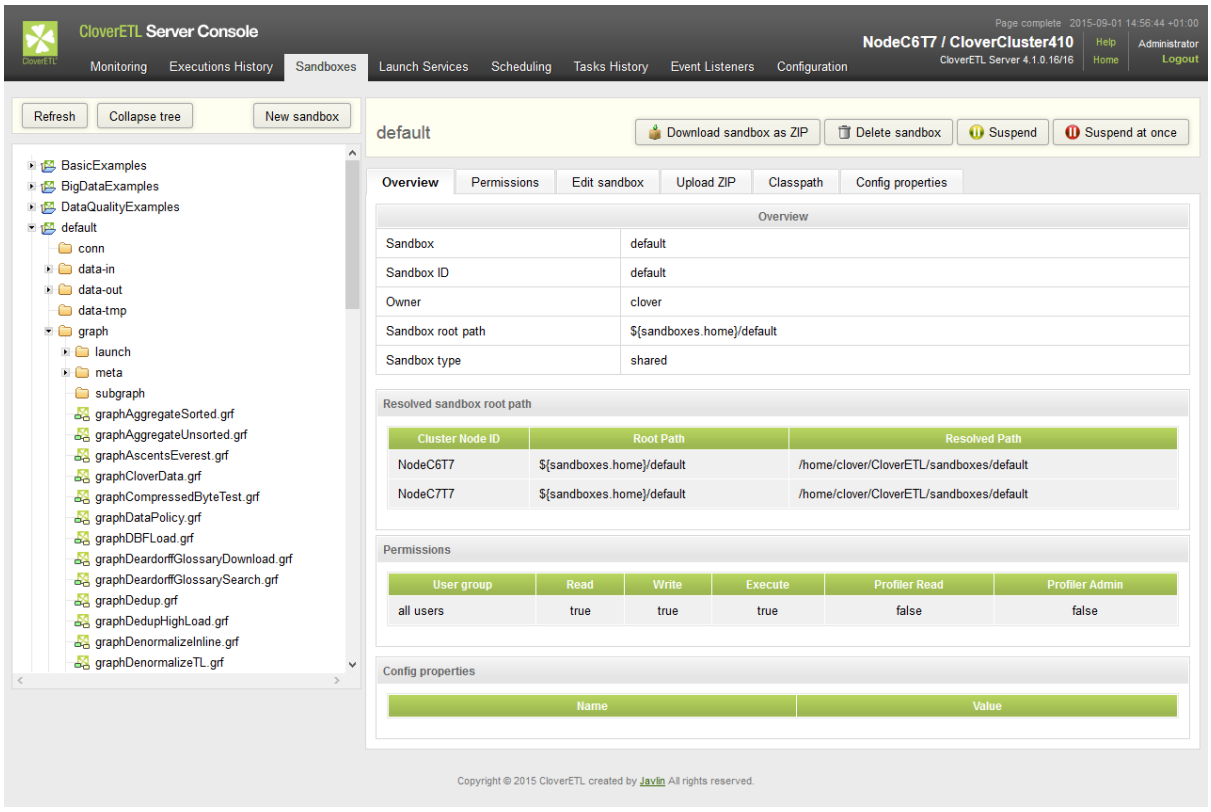


Figure 15.1. Sandboxes Section in CloverETL Server Web GUI

Each sandbox in non-cluster environment is defined by following attributes:

Table 15.1. Sandbox attributes

Sandbox ID	Unique "name" of the sandbox. It is used in server APIs to identify sandbox. It must meet common rules for identifiers. It is specified by user in during sandbox creation and it can be modified later. *Note: modifying is not recommended, because it may be already used by some APIs clients.*
Sandbox	Sandbox name used just for display. It is specified by user in during sandbox creation and it can be modified later.
Sandbox root path	Absolute server side file system path to sandbox root. It is specified by user during sandbox creation and it can be modified later. Instead of the absolute path, it's recommended to use placeholder ${sandboxes.home} which may be configurable in the CloverETL Server configuration. So e.g. for the sandbox with ID "dataReports" the specified value of the "root path" would be "${sandboxes.home}/dataReports". Default value of "sandboxes.home" config property is "${user.data.home}/CloverETL/sandboxes" where the "user.data.home" is configuration property specifying home directory of the user running JVM process - it's OS dependend). Thus on the unix-like OS, the fully resolved sandbox root path may be: "/home/clover/CloverETL/sandboxes/dataReports". See Chapter 27, Clustering features(p. 183) for details about sandboxes root path in cluster environment.
Owner	It is set automatically during sandbox creation. It may be modified later.

Referencing files from the ETL graph or Jobflow

In some components you can specify file URL attribute as a reference to some resource on the file system. Also external metadata, lookup or DB connection definition is specified as reference to some file on the filesystem. With CloverETL Server there are more ways how to specify this relation.

- Relative path

  All relative paths in your graphs are considered as relative paths to the root of the same sandbox which contains job file (ETL graph or Jobflow).

- sandbox:// URLs

  Sandbox URL allows user to reference the resource from different sandboxes with standalone CloverETL Server or the cluster. In cluster environment, CloverETL Server transparently manages remote streaming if the resource is accessible only on some specific cluster node.

  See Using a Sandbox Resource as a Component Data Source (p. 188) for details about the sandbox URLs.

Sandbox Content Security and Permissions

Each sandbox has its owner which is set during sandbox creation. This user has unlimited privileges to this sandbox as well as administrators. Another users may have access according to sandbox settings.

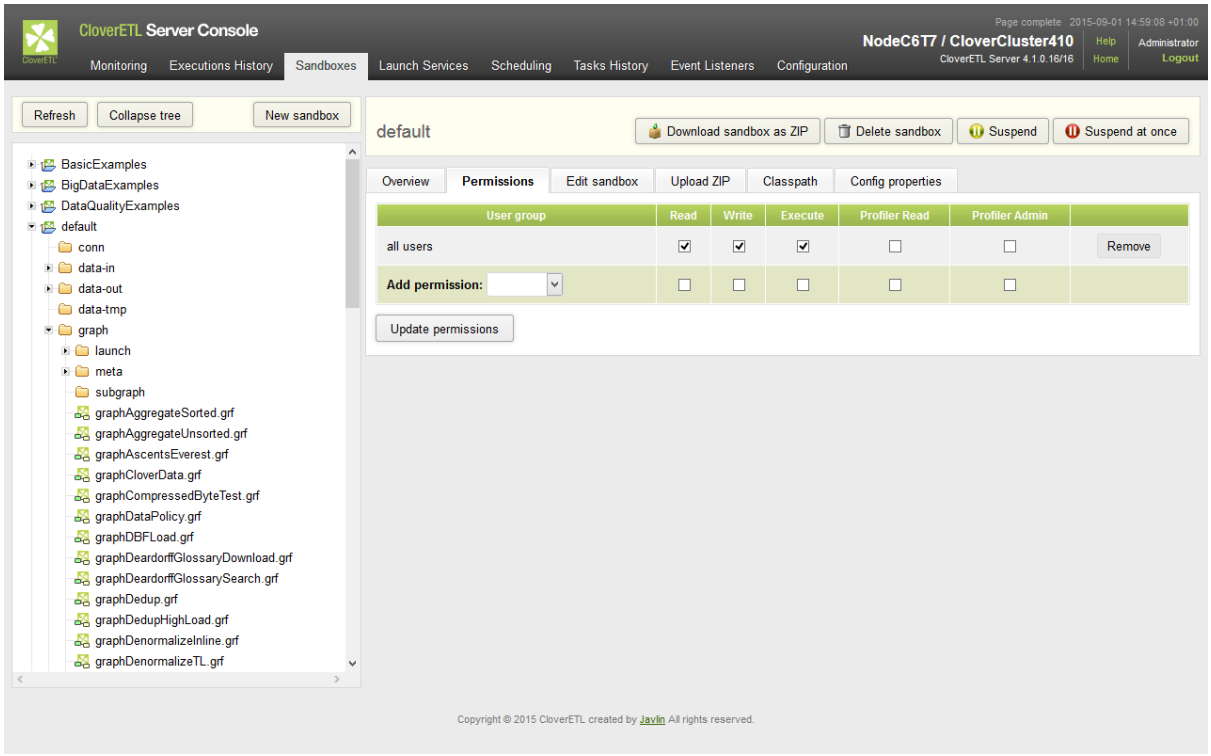


Figure 15.2. Sandbox Permissions in CloverETL Server Web GUI

Permissions to specific sandbox are modifiable in **Permissions** tab in sandbox detail. In this tab, selected user groups may be allowed to perform particular operations.

There are 3 types of operations:

Table 15.2. Sandbox permissions

Read	Users can see this sandbox in their sandboxes list.
Write	Users can modify files in the sandbox through CS APIs.
Execute	Users can execute jobs in this sandbox. *Note: jobs executed by "graph event listener" and similar features is actually executed by the same user as job which is source of event. See details in "graph event listener". Job executed by schedule trigger is actually executed by the schedule owner. See details in Chapter 20, Scheduling (p. 120). If the job needs any files from the sandbox (e.g. metadata), user also must have read permission, otherwise the execution fails.*
Profiler Read	User can view results of profiler jobs executed from the sandbox.
Profiler Admin	User can administer results of profiler jobs executed from the sandbox.

Please note that, these permissions modify access to the content of specific sandboxes. In additions, it's possible to configure permissions to perform operations with sandbox configuration. e.g. create sandbox, edit sandbox, delete sandbox, etc. Please see Chapter 14, Users and Groups (p. 86) for details.

Sandbox Content

Sandbox should contain jobflows, graphs, metadata, external connection and all related files. Files especially graph or jobflow files are identified by relative path from sandbox root. Thus you need two values to identify specific job file: sandbox and path in sandbox. Path to the Jobflow or ETL graph is often referred as "Job file".

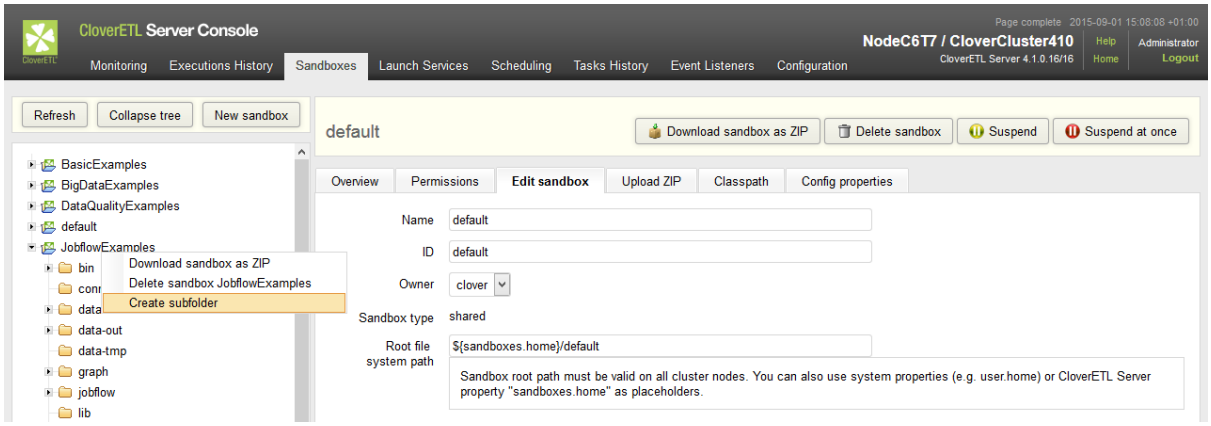


Figure 15.3. Web GUI - section "Sandboxes" - context menu on sandbox

Although web GUI section **sandboxes** isn't file-manager, it offers some useful features for sandbox management.

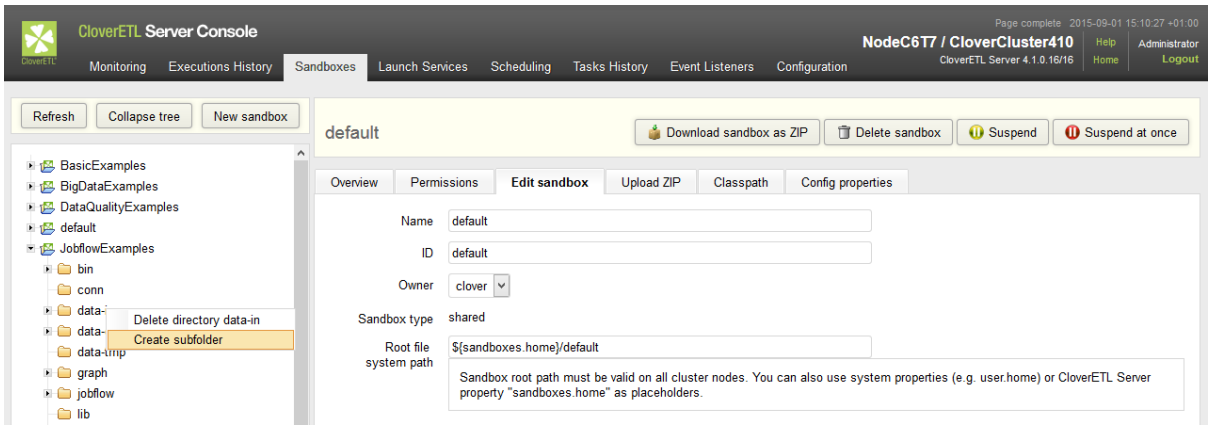


Figure 15.4. Web GUI - section "Sandboxes" - context menu on folder

Download sandbox as ZIP

Select sandbox in left panel, then web GUI displays button "Download sandbox as ZIP" in the tool bar on the right side.

Created ZIP contains all readable sandbox files in the same hierarchy as on file system. You can use this ZIP file for upload files to the same sandbox, or another sandbox on different server instance.

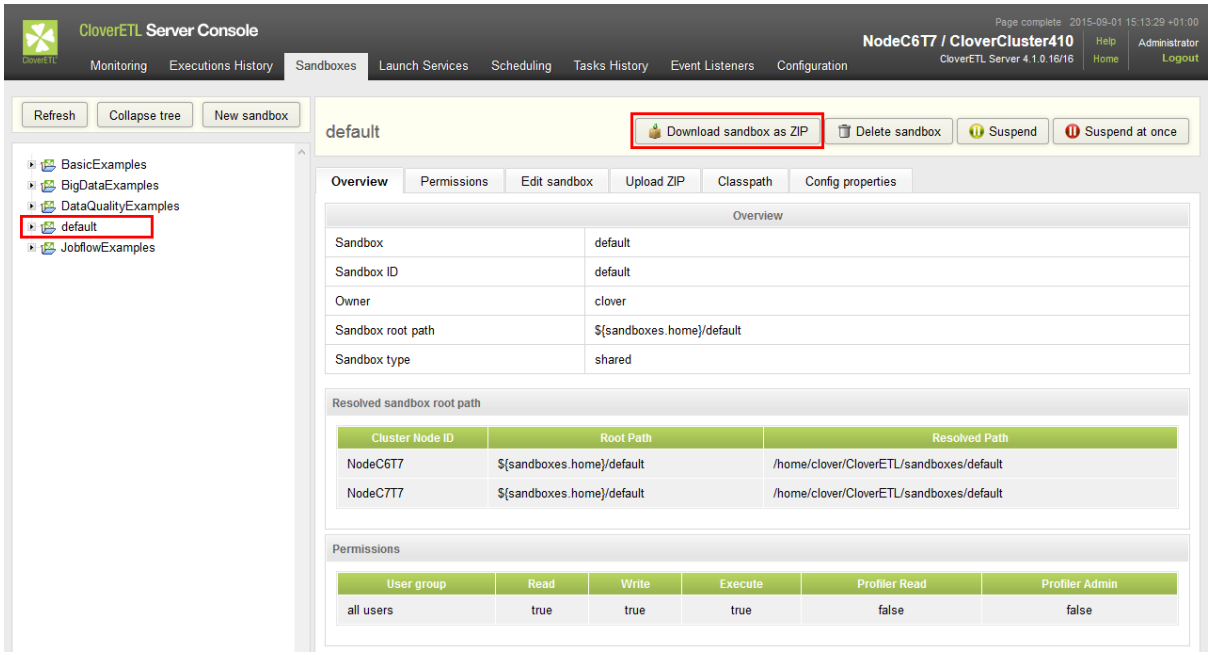


Figure 15.5. Web GUI - download sandbox as ZIP

Upload ZIP to sandbox

Select sandbox in left panel. You must have write permission to the selected sandbox. Then select tab "Upload ZIP" in the right panel. Upload of ZIP is parametrized by couple of switches, which are described below. Open common file chooser dialog by button "+ Upload ZIP". When you choose ZIP file, it is immediately uploaded to the server and result message is displayed. Each row of the result message contains description of one single file upload. Depending on selected options, file may be skipped, updated, created or deleted.

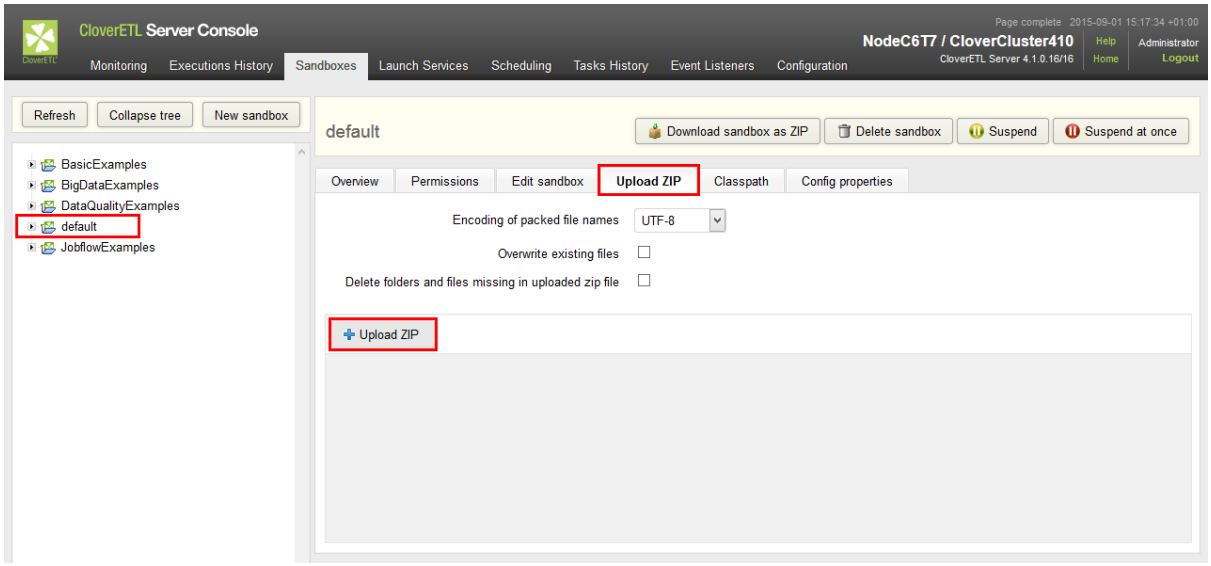


Figure 15.6. Web GUI - upload ZIP to sandbox

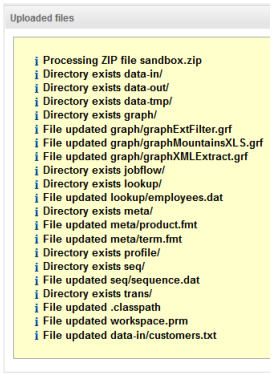


Figure 15.7. Web GUI - upload ZIP results

Table 15.3. ZIP upload parameters

Label	Description
Encoding of packed file names	File names which contain special characters (non ASCII) are encoded. By this select box, you choose right encoding, so filenames are decoded properly.
Overwrite existing files	If this switch is checked, existing file is overwriten by new one, if both of them are stored in the same path in the sandbox and both of them have the same name.
Replace content	If this option is enabled, all files which are missing in uploaded ZIP file, but they exist in destination sandbox, will be deleted. This option might cause loose of data, so user must have special permission "May delete files, which are missing in uploaded ZIP" to enable it.

Download file in ZIP

Select file in left panel, then web GUI displays button "Download file as ZIP" in the tool bar on the right side.

Created ZIP contains just selected file. This feature is useful for large files (i.e. input or output file) which cannot be displayed directly in web GUI. So user can download it.

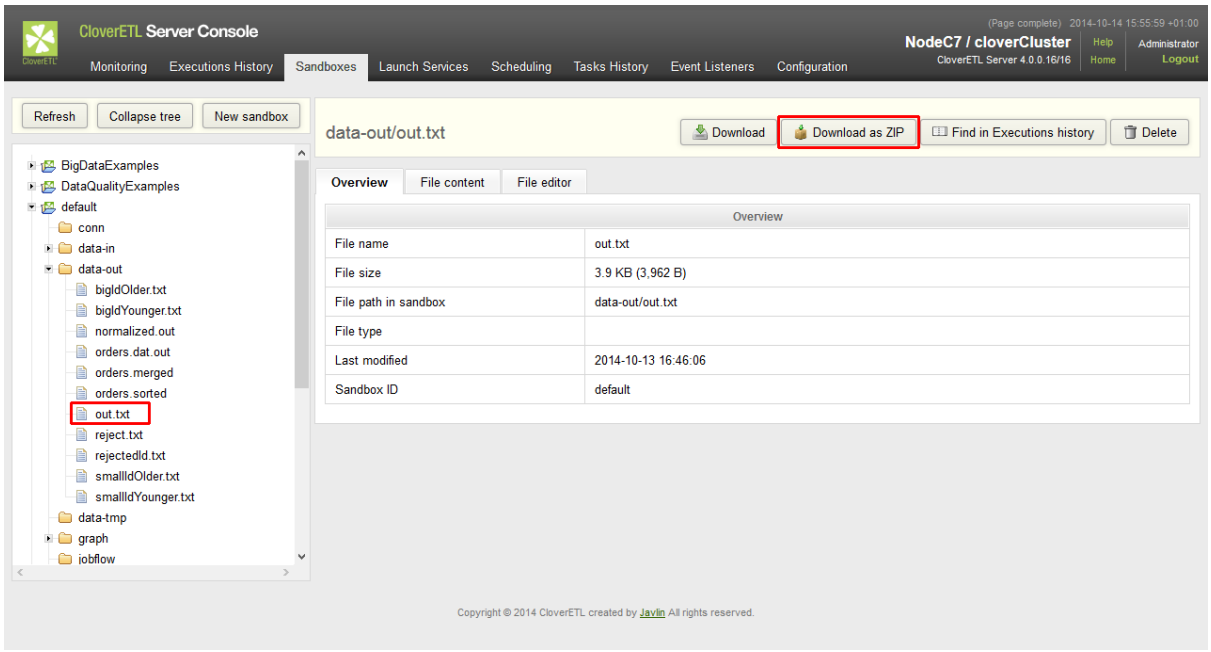


Figure 15.8. Web GUI - download file as ZIP

Download file HTTP API

It is possible to download/view sandbox file accessing "download servlet" by simple HTTP GET request:

```
http://[host]:[port]/[Clover Context]/downloadFile?[Parameters]
```

Server requires BASIC HTTP Authentication. Thus with linux command line HTTP client "wget" it would look like this:

```
wget --user=clover --password=clover
    http://localhost:8080/clover/downloadFile?sandbox=default\&file=data-out/data.dat
```

Please note, that ampersand character is escaped by back-slash. Otherwise it would be interpreted as command-line system operator, which forks processes.

URL Parameters

- sandbox - Sandbox code. Mandatory parameter.
- file - Path to the file relative from sandbox root. Mandatory parameter.
- zip - If set to "true", file is returned as ZIP and response content type is "application/x-zip-compressed". By default it is false, so response is content of the file.

Job config properties

Each ETL graph or Jobflow may have set of config properties, which are applied during the execution. Properties are editable in web GUI section "sandboxes". Select job file and go to tab "Config properties".

The same config properties are editable even for each sandbox. Values specified for sandbox are applied for each job in the sandbox, but with lower priority then config properties specified for the job.

If neither sandbox or job have config properties specified, defaults from main server configuration are applied. Global config properties related to Job config properties have prefix "executor.". E.g. server property "executor.classpath" is default for Job config property "classpath". (See Part III, “Configuration” (p. 40) for details)

In addition, it is possible to specify additional job parameters, which can be used as placeholders in job XML. Please keep in mind, that these placeholders are resolved during loading and parsing of XML file, thus such job couldn't be pooled.

If you use a relative path, the path is relative to `${SANDBOX_ROOT}`.

In path definition, you can use system properties - e.g. `${java.io.tmpdir}` - and some of server config properties: `${sandboxes.home}`, `${sandboxes.home.partitioned}` and `${sandboxes.home.local}`.

Table 15.4. Job config parameters

Property name	Default value	Description
tracking_interval	2000	Interval in ms for sampling nodes status in running transformation.
max_running_concurrently	unlimited	Max number of concurrently running instances of this transformation.
enqueue_executions	false	Boolean value. If it is true, executions above max_running_concurrently are enqueued, if it is false executions above max_running_concurrently fail.
log_level	INFO	Log4j log level for this graph executions. (ALL | TRACE | DEBUG | INFO | WARN | ERROR

Property name	Default value	Description
		| FATAL) For lower levels (ALL, TRACE or DEBUG), also root logger level must be set to lower level. Root logger log level is INFO by default, thus transformation run log does not contain more detail messages then INFO event if job config parameter "log_level" is set properly. See Chapter 11, Logging (p. 74) for details about log4j configuration.
max_graph_instance_age	0	Time interval in ms which specifies how long may transformation instance last in server's cache. 0 means that transformation is initialized and released for each execution. Transformation cannot be stored in the pool and reused in some cases (transformation uses placeholders using dynamically specified parameters)
classpath		List of paths or jar files which contain external classes used in the job file (transformations, generators, JMS processors). All specified resources will be added to runtime classpath of the transformation job. All Clover Engine libraries and libraries on application-server's classpath are automatically on the classpath. Separator is specified by Engine property "DEFAULT_PATH_SEPARATOR_REGEX". Directory path must always end with slash character "/", otherwise ClassLoader doesn't recognize it's a directory. Server always automatically adds "trans" subdirectory of job's sandbox, so It doesn't have to be added explicitly.
compile_classpath		List of paths or jar files which contain external classes used in the job file (transformations, generators, JMS processors) and related libraries for their compilation. Please note, that libraries on application-server's classpath aren't included automatically. Separator is specified by Engine property "DEFAULT_PATH_SEPARATOR_REGEX". Directory path must always end with slash character "/", otherwise ClassLoader doesn't recognize it's a directory. Server always automatically adds "SANDBOX_ROOT/trans/" directory and all JARs in "SANDBOX_ROOT/lib/" directory, so they don't have to be added explicitly.
classloader_caching	false	Clover creates new classloaders whenever is necessary to load a class in runtime. For example Reformat component with a Java transformation has to create new classloader to load the class. It is worth noting that classloaders for JDBC drivers are not re-created. Classloader cache is used to avoid PermGen out of memory errors (some JDBC drivers automatically register itself to DriverManager, which can cause the classloader cannot be released by garbage collector). This behaviour can be inconvenient for example if you want to share POJO between components. For example, a Reformat component creates an object (from a jar file on runtime classpath) and stores it into dictionary. Another Reformat component get

Property name	Default value	Description
		the object from the dictionary and tries to cast the object to expected class. ClassCastException is thrown due different classloaders used in the Reformat components. Using this flag you can force CloverServer to re-use classloader when possible.
skip_check_config	default value is taken from engine property	Switch which specifies whether check config must be performed before transformation execution.
password		This property is deprecated. Password for decoding of encoded DB connection passwords.
verbose_mode	true	If true, more descriptive logs of job runs are generated.
use_jmx	true	If true, job executor registers jmx mBean of running transformation.
debug_mode	false	If true, edges with debug enabled will store data into files in a debug directory. Without explicit setting, running of a graph from Designer with server integration would set the debug_mode to true. On the other hand, running of a graph from the server console sets the debug_mode to false.
delete_obsolete_temp_files	false	If true, system will remove temporary files produced during previous finished runs of respective job. This property is useful together with enabled debug mode ensuring that obsolete debug files from previous runs of a job are removed from temp space. This property is set to "true" by default when executing job using designer-server integration.
use_local_context_url	false	If true, the context URL of a running job will be a local "file:" URL. Otherwise, a "sandbox:" URL will be used.
jobflow_token_tracking	true	If false, token tracking in jobflow executions will be disabled.
locale	DEFAULT_LOCALE engine property	Can be used to override the DEFAULT_LOCALE engine property.
time_zone	DEFAULT_TIME_ZONE engine property	Can be used to override the DEFAULT_TIME_ZONE engine property.

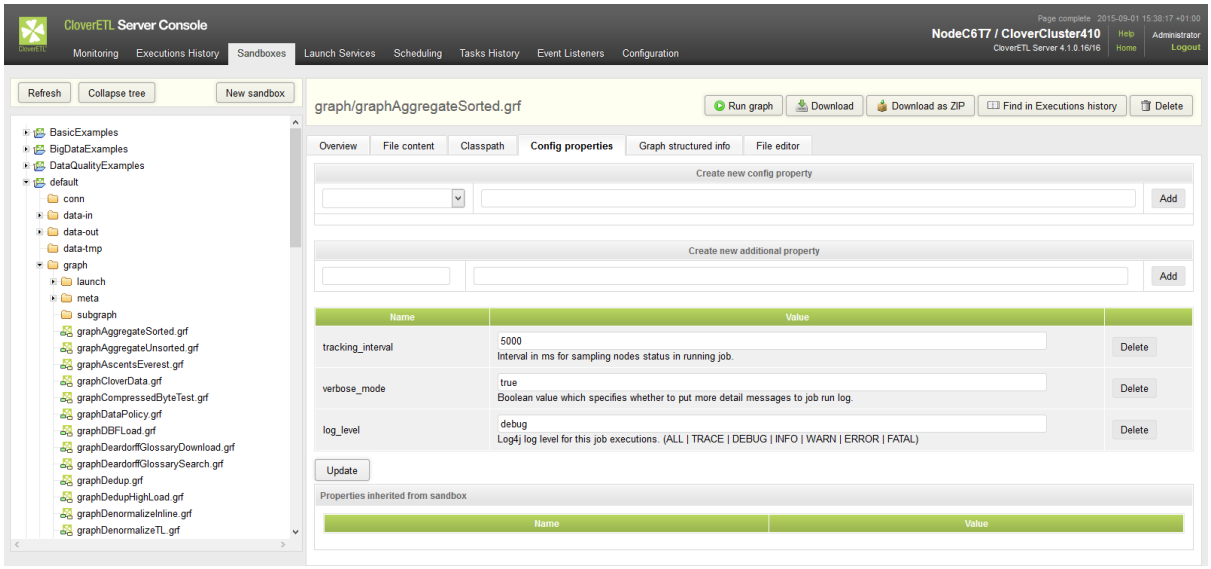


Figure 15.9. Job config properties

WebDAV access to sandboxes

Since 3.1

WebDAV API allows you to access and manage sandbox content using a standard WebDAV specification.

Specifically, it allows for:

- Browsing a directory structure
- Editing files
- Removing files/folders
- Renaming files/folders
- Creating files/folders
- Copying files
- Moving files

The WebDAV interface is accessible from the URL: "http://[host]:[port]/clover/webdav".

Note: Although common browsers will open this URL, most of them are not rich WebDAV clients. Thus, you will only see a list of items, but you cannot browse the directory structure.

WebDAV clients

There are many WebDAV clients for various operating systems, some OS support WebDAV natively.

Linux like OS

Great WebDAV client working on linux systems is Konqueror. Please use different protocol in the URL: `webdav://[host]:[port]/clover/webdav`

Another WebDAV client is Nautilus. Use different protocol in the URL `dav://[host]:[port]/clover/webdav`.

MS windows

Last distributions of MS Windows (Win XP and later) have native support for WebDAV. Unfortunatelly, it is more or less unreliable, so it is recommended to use some free or commercial WebDAV client.

- The best WebDAV client we've tested is BitKinex: http://www.bitkinex.com/webdavclient
- Another option is to use Total Commander (http://www.ghisler.com/index.htm) with WebDAV plugin: http://www.ghisler.com/plugins.htm#filesys

Mac OS

Mac OS supports WebDAV natively and in this case it should be without any problems. You can use "finder" application, select "Connect to the server ..." menu item and use URL with HTTP protocol: "http://[host]:[port]/clover/webdav".

WebDAV authentication/authorization

CloverETL Server WebDAV API uses the HTTP Basic Authentication by default. However it may be reconfigured to use HTTP Digest Authentication. Please see Part III, “Configuration” (p. 40) for details.

Digest Authentication may be useful, since some WebDAV clients can't work with HTTP Basic Authentication, only with Digest Authentication.

HTTP Digest Authentication is feature added to the version 3.1. If you upgraded your older CloverETL Server distribution, users created before the upgrade cannot use the HTTP Digest Authentication until they reset their passwords. So when they reset their passwords (or the admin does it for them), they can use Digest Authentication as well as new users.

Chapter 16. CloverETL Server Monitoring

Monitoring section in the server Web GUI displays useful information about current performance of the standalone CloverETL Server or all cluster nodes if the clustering is enabled.

Monitoring section of the standalone server has slightly different design from cluster env. Basically in case of standalone server, the server-view is the same as node detail in cluster env.

The section is refreshed each 15 seconds so the displayed data is up-to-date. Page can be also anytime refreshed manually by the "Refresh" button

Standalone server detail

Standalone server detail view displays info collected from the standalone server grouped in several panels:

- Status description - Hidden by default, but when displayed, there is a text description of the server status.
- Performance - Figure showing two basic performance values gathered by default couple of last minutes. The interval may be configurable by config property "cluster.node.sendinfo.history.interval". Table on the right shows some more performance related values.
- Environment - Attributes describing environment of CloverETL Server, JVM and the OS
- 10 longest-running jobs - List of jobs currently running, however just 10 "oldest" runs are displayed.
- Status - Node statuses history since the server restart.

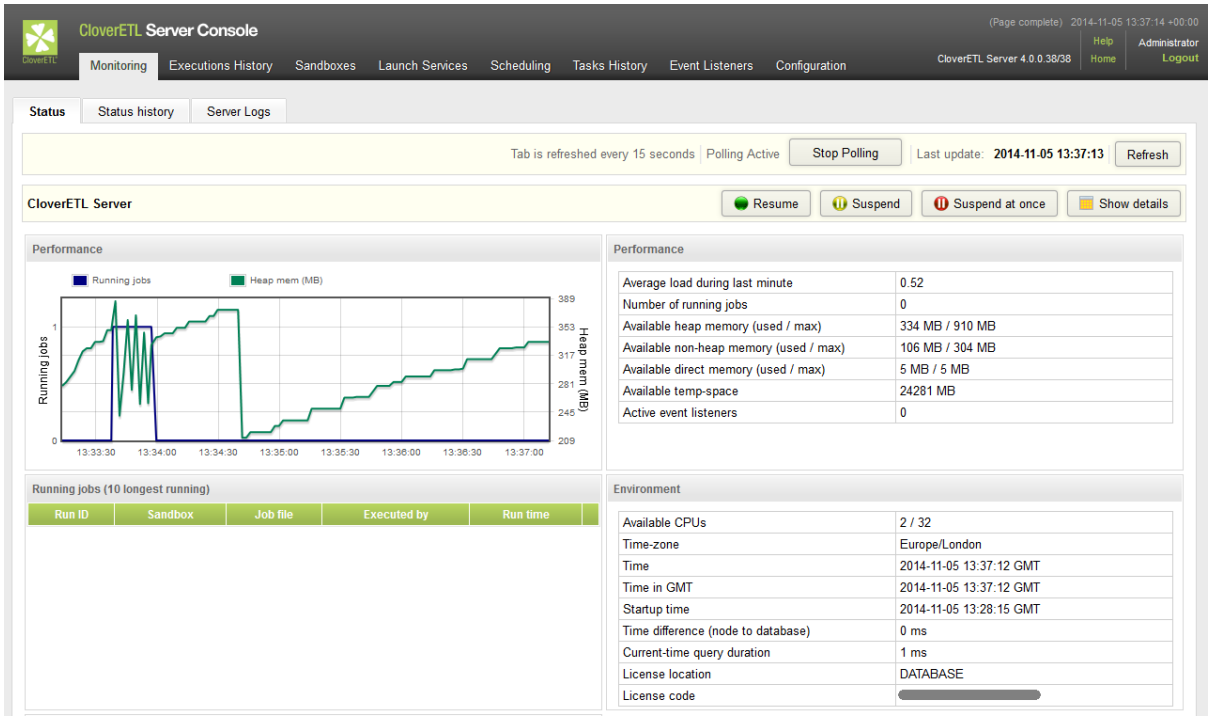


Figure 16.1. Standalone server detail

Cluster overview

Cluster overview displays info collected from all cluster nodes grouped in several panels:

- List of nodes with toolbar - allows manipulation with selected nodes
- Status history - Displays last 10 status changes for all cluster nodes
- Node detail - Displays several basic performance attributes for selected nodes. It's visible on the right side only when activated by button on the toolbar.
- Running jobs - It's displayed only when there are running jobs.

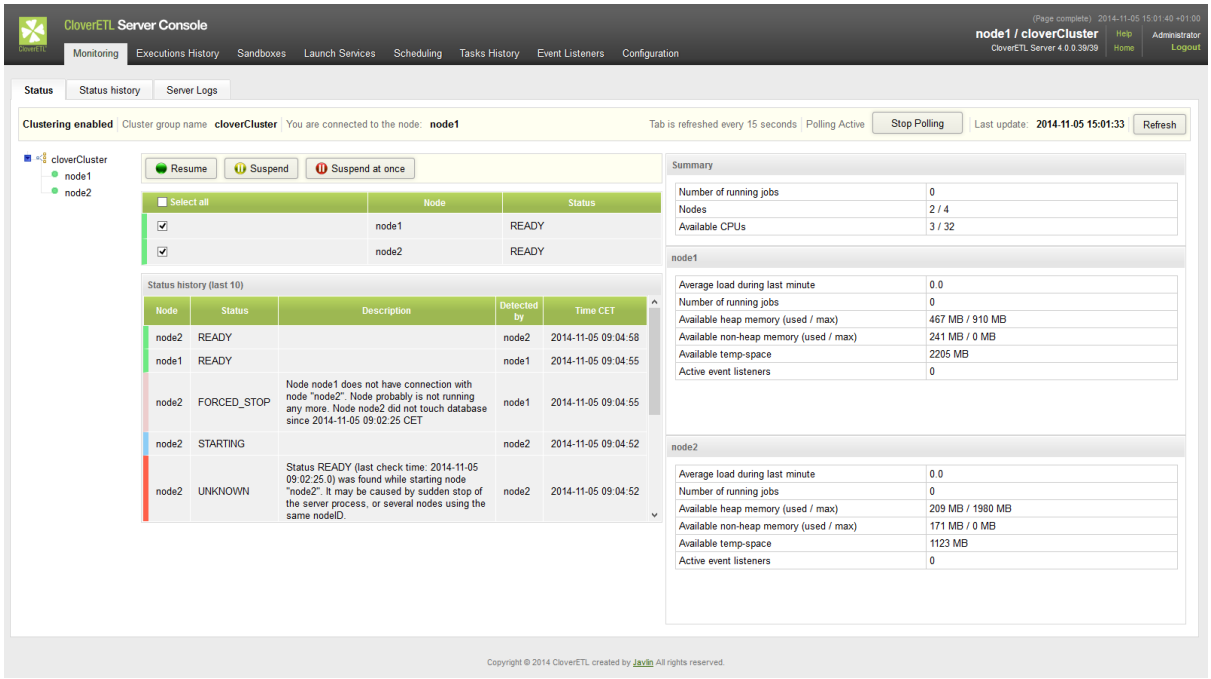


Figure 16.2. Cluster overview

Node detail

Basically it's similar to the "Standalone server detail" mentioned above, however it displays detail info about node selected in the tree on the left.

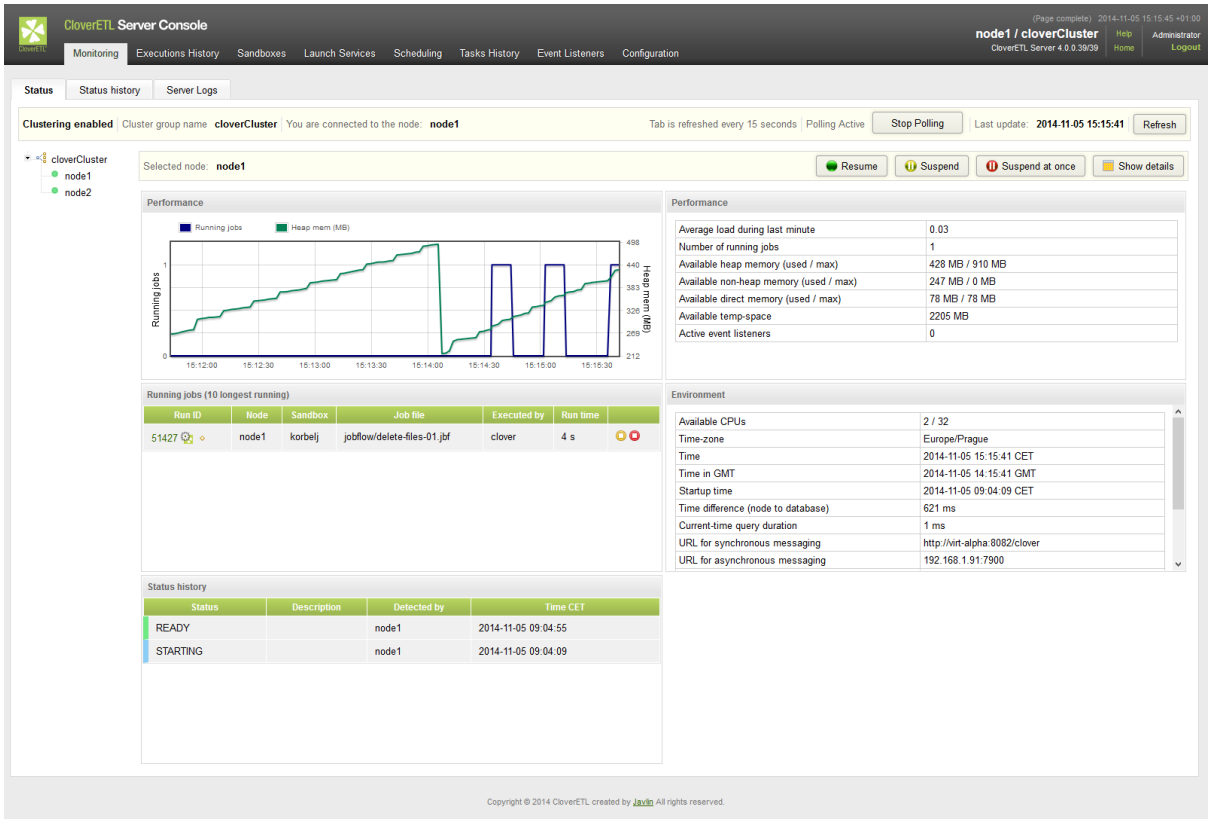


Figure 16.3. Node detail

Server Logs

Tab Server Logs allows user to investigate log messages logged on other cluster nodes. Since the log messages are collected in memmory, the maximum of collected messages is relatively low by default, however it's customisable.

There are also 3 different "Log types":

- COMMON - Ordinary server logs as stored in log files.
- CLUSTER - Only cluster - related messages are visible in this log
- LAUNCH_SERVICES - Only requests for launch services
- AUDIT - Detail logging of operations called on the CloverETL Server core. Since the full logging may affect server performance, it's disabled by default. See Server Audit logs (p. 75) for details
- USER_ACTION - Contains some of user operations, e.g. login, logout, job execution

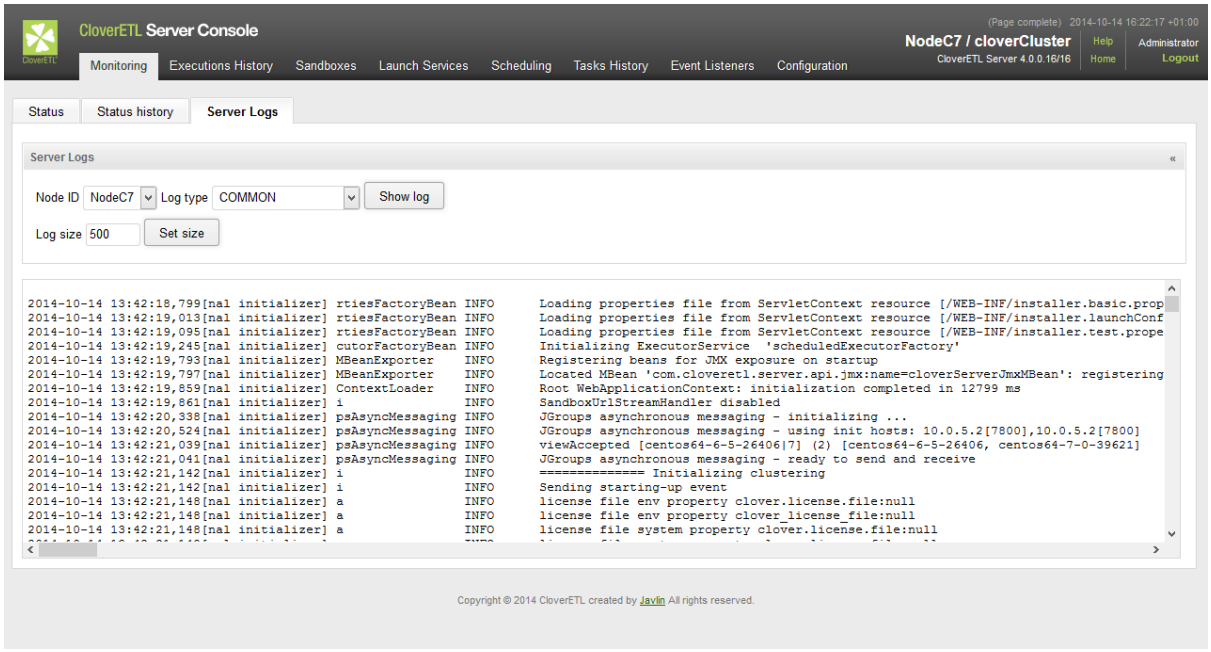


Figure 16.4. Server Logs

Chapter 17. Server Configuration Migration

CloverETL Server provides means to migrate its configuration (e.g. event listeners, schedules etc.) or parts of the configuration between separate instances of the server. A typical use case is deployment from test environment to production - this involves not only deployment of CloverETL graphs, but also copying parts of configuration such as file event listeners etc.

Configuration migration is performed in 2 steps - export of the configuration from the source server, followed by import of the configuration at the destination server. After exporting, the configuration is stored as an XML file. The file can be modified maunally before import, for example to migrate only parts of the configuration. Additionally, the configuration file can be stored in a versioning system (such as Subversion or Git) for versioning of the CloverETL Server configuration.

It is recommended to perform import of configuration on a suspended CloverETL Server and to plan for maintenance. Additionally, it is recommended to backup the CloverETL Server configuration database before import.

The following items are parts of the *Server Configuration* and can be migrated between servers:

- Users & Groups (see Chapter 14, Users and Groups (p. 86))
- Sandboxes (see Chapter 15, Server Side Job files - Sandboxes (p. 95))
- Job Parameters (see Chapter 18, Graph/Jobflow parameters (p. 116))
- Schedules (see Chapter 20, Scheduling (p. 120))
- Event Listeners (see Graph Event Listeners (p. 137), Jobflow Event Listeners (p. 145), JMS messages listeners (p. 147), File event listeners (p. 152))
- Launch Services (see Launch Services (p. 169))
- Temp Spaces (see Chapter 12, Temp Space Management (p. 77))

Permissions for Configuration Migration

Whether user is entitled to perform configuration migration is determined by having *Server Configuration Managament* permission; this permission has two sub-permissions: *Export Server Configuration* and *Import Server Configuration* (see Group Permissions (p. 94) for further information on permissions). These permissions are of higher priority than permissions related to particular migrated item type - so even if the user does not have a permission e.g. to list server's schedules, with *Export Server Configuration* he will be allowed to export all of defined schedules. The same is true for adding and changing items with the *Import Server Configuration* permission.

Server Configuration Export

Export of server configuration is performed from the Server Console - the screen for export can be found in section **Configuration** > **Export**. You can choose which items will be exported (see Figure 17.1 (p. 111)). After clicking on the **Export Configuration** an XML file will be offered for download. The name of the XML reflects time when the configuration was exported.

In case user manually edits the exported XML file it is important to ensure its validity. This can be done by validation against XSD schema. The schema for configuration XML document can be found at `http://[host]:[port]/[contextPath]/schemas/clover-server-config.xsd`.

The XML file contains selected items of the CloverETL server instance. The file can by modified before import to another server instance - for example to import schedules only.

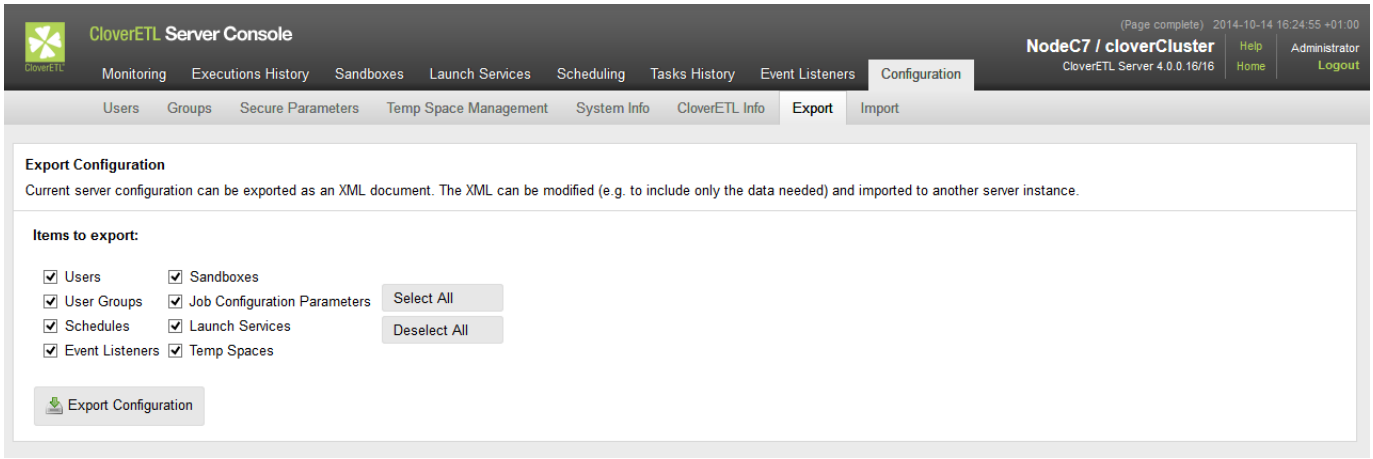


Figure 17.1. Server Configuration Export screen

Server Configuration Import

Import of CloverETL Server configuration merges configuration exported from another server into the running server instance where the import was initiated. The configuration to be imported is loaded from an XML file created by export, see Server Configuration Export (p. 110). Import of server configuration is performed from the Server Console - the screen for import can be found in section **Configuration** > **Import**.

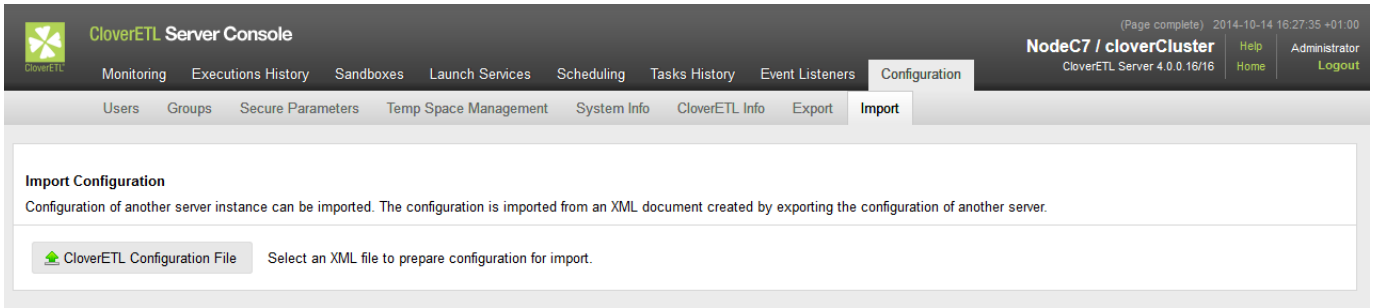


Figure 17.2. Server Configuration Import screen

The XML configuration file defines configuration items to be imported into the destination server. The items are matched against current configuration of the destination server. Depending on result of the matching, the items from the XML configuration are either added to the destination server or will update existing item with properties defined in the XML file. Matching of items is based on a key that depends on the item type:

- users - user code
- user groups - group code
- sandboxes - sandbox code
- job parameters - triplet (job parameter name, sandbox code, job file)
- event listeners - event listener name
- schedule - schedule description
- launch service - triplet (service name, server user, service user group)
- temp spaces - pair (temp space node ID, temp space path)

Configuration Import Process

Uploading Configuration

The first step in the configuration import is to upload the XML file to the CloverETL server. After clicking on **CloverETL Configuration File** button a window is opened where user can select an XML file with the

configuration to import. The file is uploaded automatically after the dialog is closed. Once upload is finished the name of the uploaded file is shown in the toolbar along with **CloverETL Configuration File** button. In case reading of configuration from XML has finished without error, additional controls are displayed in the toolbar:

- **Preview Import** button to perform "dry run" of the configuration import
- **Commit Import** button to perform actual import of configuration to server's database
- **Import Options** section to further configure import process:
  - **New only** option specifies that only new items will be imported leaving existing items on server untouched
- **Import Items** section to select what item types will be imported to the server

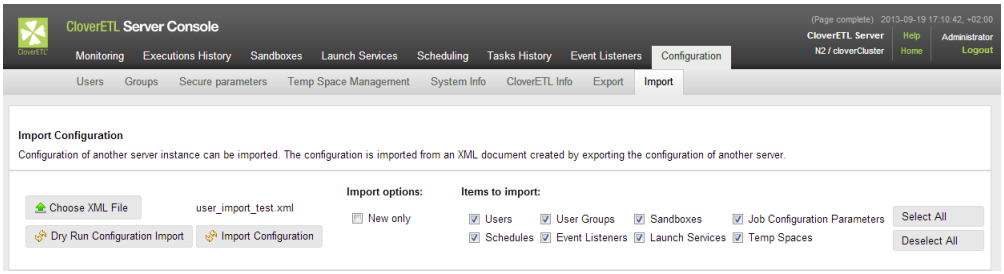


Figure 17.3. Server Configuration uploaded

Note

When transfering configuration from one server instance to another, it is important that these instances are of compatible, preferably the same, version. The user is notified when the source and target differ at at least minor version number (e.g. 3.4.1 and 3.5.0). It is also recommended not to transfer configuration between clustered and non-clustered server instances.

Verifying Configuration

Once the configuration is uploaded, the system executes "dry run" of the configuration import automatically. The *dry run* performs configuration import, but no data are actually written do the server's database. The outcome of the operation is **Import Log** with messages related either to the configuration as a whole or to particular imported items (see Figure 17.4 (p. 113)). There is also another view of **Imported Items** to list all added/updated items grouped into tables according to their types. Each item has an icon indicating result of the item import operation:

- - the item has been added as a new one
- - the item has been updated
- - the item has been updated, but none of its properties has changed
- - the item has been removed

For the updated items, the state of the item before update is shown in the lower of the rows with less contrast text color, the new values of item's properties are shown in upper of the rows with regular text color. Actual changes are highlighted by background color change on respective property and also on both rows. The **Imported Items** view can be filtered using radio buttons above it:

- **Changes only** button will display only items that have been either added or actually changed by update
- **All updates** button will display all of imported items, event those identic to already present ones

Example 17.1. Example of simple configuration defining one new server user.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cloverServerConfiguration xmlns="http://cloveretl.com/server/data" timeZone="Europe/Berlin">
    <usersList>
        <user disabled="false">
            <username>johnsmith</username>
            <firstName>John</firstName>
            <lastName>Smith</lastName>
            <email>smithj@tnet.com</email>
            <domain>clover</domain>
            <password>SITx8e1pjoo4em17As3LNw==</password>
            <passwordA1Digest>70151bae7488da4159ac5ccec97d0995</passwordA1Digest>
            <userGroups>
                <groupCode>all_users</groupCode>
                <groupCode>job_managers</groupCode>
            </userGroups>
        </user>
    </usersList>
</cloverServerConfiguration>
```

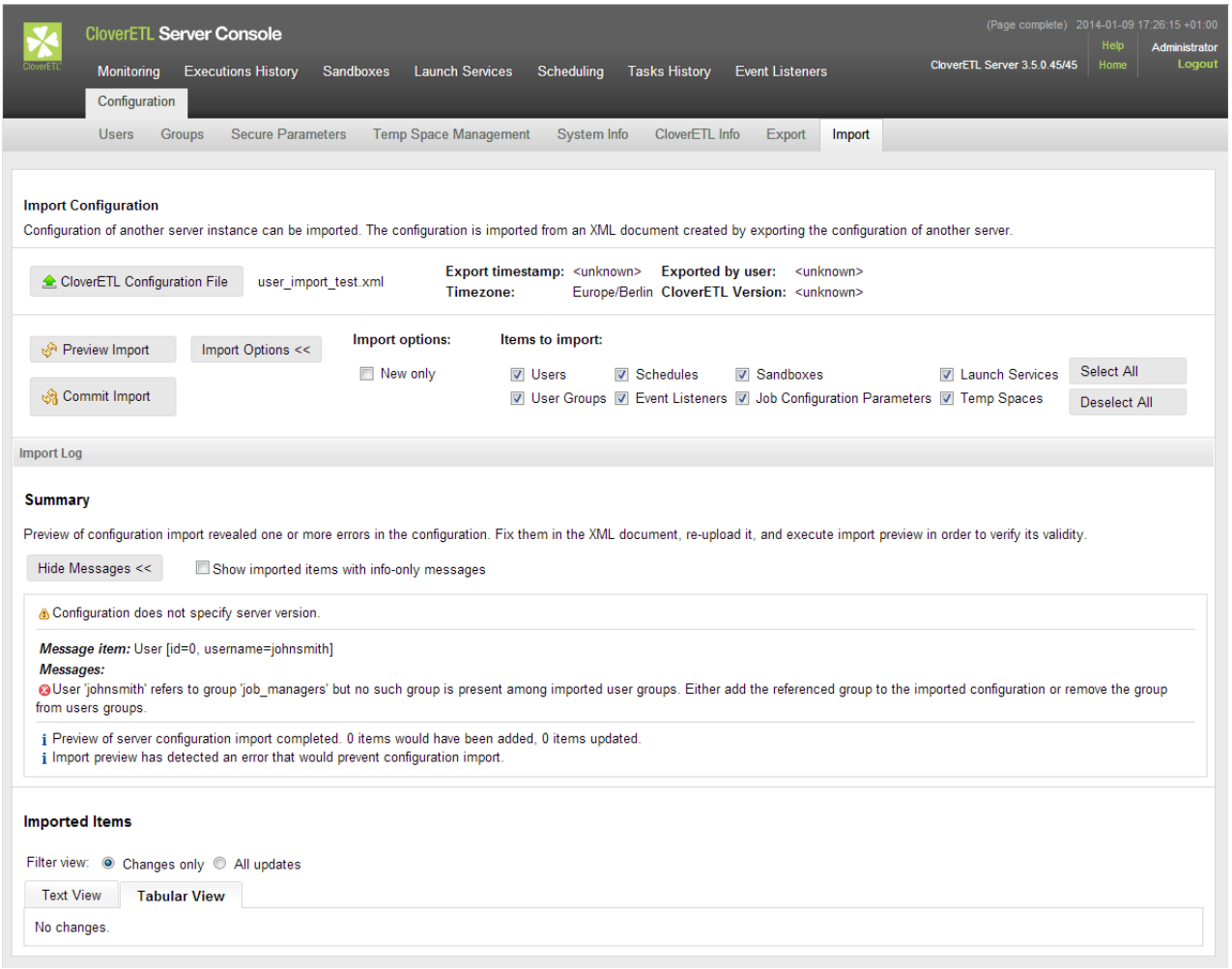


Figure 17.4. Outcome of the import preview for configuration from Example 17.1 (p. 113)

The **Summary** in the **Import Log** says whether the dry run was successful. Should there by any problems with items imported, the item is displayed along with the cause of the error (see Figure 17.4 (p. 113)) .

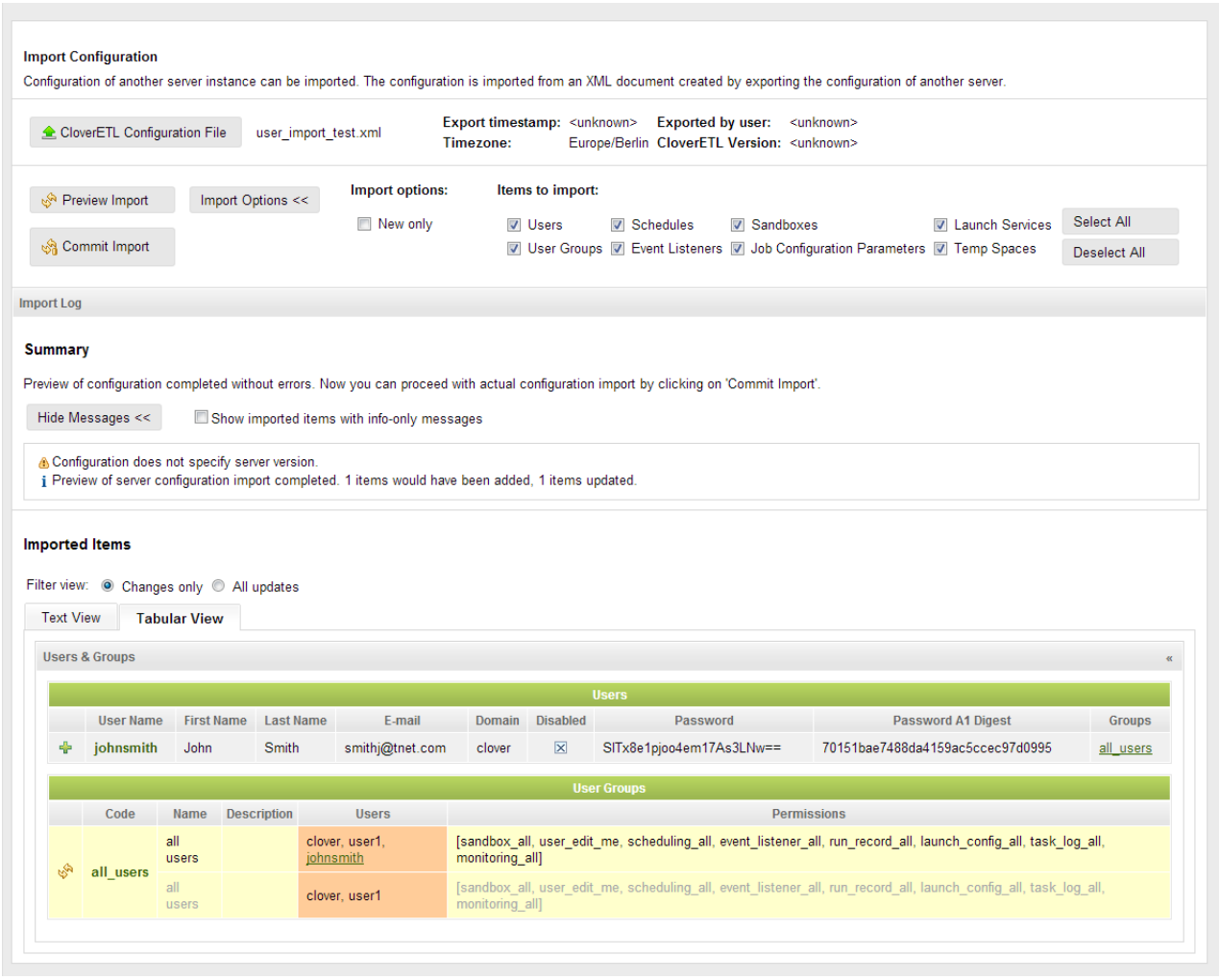


Figure 17.5. Outcome of import preview for configuration after fixing by removal of broken group reference.

User is expected follow the advices displayed in the **Import Log** and edit the XML until import preview has finished without errors.

Commiting Import

Once the import preview has finished without problems, one can proceed with actual configuration import. This is performed by clicking on the **Commit Import** button. After confirmation, **Import Log** will display outcome of the operation.

It is possible that some items will not be initialized properly after they have been imported (e.g. their initialization requires presence of a cluster node that went down in the meantime). User is notified about these problems in **Import Log** with link to the problematic item. One should check such items in appropriate section of the CloverETL Server console and change their settings to fix the issue or remove them.

Part V. Using Graphs

Chapter 18. Graph/Jobflow parameters

The CloverETL Server passes a set of parameters to each graph or jobflow execution.

Keep in mind that placeholders (parameters) ${paramName} are resolved only during the initialization (loading of XML definition file), so if you need the parameters to be resolved for each job execution, you cannot set the job to be pooled. However, current parameter values are always accessible by inline Java code like this:

```
String runId = getGraph().getGraphProperties().getProperty("RUN_ID");
```

Properties may be added or replaced like this:

```
getGraph().getGraphProperties().setProperty("new_property", value );
```

This is set of parameters which are always set by CloverETL Server:

Table 18.1. Defaults for graph execution configuration - see section Graph config properties for details

key	description
SANDBOX_CODE	Code of sandbox which contains executed graph.
JOB_FILE	Path to the file, relative to sandbox root path.
SANDBOX_ROOT	Absolute path sandbox root.
RUN_ID	ID of the graph execution. In standalone mode or in cluster mode, it is always unique. It may be lower then 0 value, if the run record isn't persistent. See Launch Services (p. 169) for details.
PARENT_RUN_ID	Run ID of the graph execution which is parent to the current one. Useful when the execution is subgraph, child-job of some jobflow or worker for distributed transformation in cluster. When the execution doesn't have parent, the PARENT_RUN_ID is the same as RUN_ID.
ROOT_RUN_ID	Run ID of the graph execution which is root execution to the current one (the one which doesn't have parent). Useful when the execution is subgraph, child-job of some jobflow or worker for distributed transformation in cluster. When the execution doesn't have parent, the ROOT_RUN_ID is the same as RUN_ID.
CLOVER_USERNAME	Username of user who launched the graph or jobflow.
NODE_ID	Id of node running the graph or jobflow.

Parameters by execution type

Additional parameters are passed to the graph depending on how the graph is executed.

Executed from Web GUI

no more parameters

Executed by Launch Service invocation

Service parameters which have attribute **Pass to graph** enabled are passed to the graph not only as "dictionary" input data, but also as graph parameter.

Executed by HTTP API run graph operation invocation

Any URL parameter with "param_" prefix is passed to executed graph but without "param_" prefix. i.e. "param_FILE_NAME" specified in URL is passed to the graph as property named "FILE_NAME".

Executed by RunGraph component

Since 3.0 only parameters specified by "paramsToPass" attribute are passed from the "parent" graph to the executed graph. However common properties (RUN_ID, PROJECT_DIR, etc.) are overwritten by new values.

Executed by WS API method executeGraph invocation

Parameters with values may be passed to the graph with the request for execution.

Executed by task "graph execution" by scheduler

Table 18.2. passed parameters

key	description
EVENT_SCHEDULE_EVENT_TYPE	Type of schedule SCHEDULE_PERIODIC | SCHEDULE_ONETIME
EVENT_SCHEDULE_LAST_EVENT	Date/time of previous event
EVENT_SCHEDULE_DESCRIPTION	Schedule description, which is displayed in web GUI
EVENT_USERNAME	User who "owns" the event. For schedule it is the user who created the schedule
EVENT_SCHEDULE_ID	ID of schedule which triggered the graph

Executed from JMS listener

There are many graph parameters and dictionary entries passed, depending on the type of incomming message. See details in JMS messages listeners (p. 147).

Executed by task "Start a graph" by graph/jobflow event listener

Since 3.0 only specified properties from "source" job are passed to executed job by default. There is server config property "graph.pass_event_params_to_graph_in_old_style" which can change this behavior so that ALL parameters from "source" job are passed to the executed job. This switch is implemented for backwards compatibility. Regarding the default behaviour: You can specified list of parameters to pass in the editor of graph event listener. Please see Task - Execution of Graph (p. 125) for details.

However following parameters with current values are always passed to the target job

Table 18.3. passed parameters

key	description
EVENT_RUN_SANDBOX	Sandbox with graph, which is source of the event

key	description
EVENT_JOB_EVENT_TYPE	GRAPH_STARTED | GRAPH_FINISHED | GRAPH_ERROR | GRAPH_ABORTED | GRAPH_TIMEOUT | GRAPH_STATUS_UNKNOWN, analogically JOBFLOW_* for jobflow event listeners.
EVENT_RUN_JOB_FILE	jobFile of the job, which is source of the event
EVENT_RUN_ID	ID of the graph execution, which is source of the event.
EVENT_TIMEOUT	Number of miliseconds which specifies interval of timeout. Makes sence only for "timeout" graph event.
EVENT_RUN_RESULT	Result (or current status) of the execution, which is source of the event.
EVENT_USERNAME	User who "owns" the event. For graph events it is the user who created the graph event listener

Executed by task "graph execution" by file event listener

Table 18.4. passed parameters

key	description
EVENT_FILE_PATH	Path to file, which is source of the event. Does not contain file name. Does not end with file separator.
EVENT_FILE_NAME	Filename of the file which is source of the event.
EVENT_FILE_EVENT_TYPE	SIZE | CHANGE_TIME | APPEARANCE | DISAPPEARANCE
EVENT_FILE_PATTERN	Pattern specified in file event listener
EVENT_FILE_LISTENER_ID	
EVENT_USERNAME	User who "owns" the event. For file events it is the user who created the file event listener

How to add another graph parameters

Additional "Graph Config Parameters"

It is possible to add so-called additional parameters in Web GUI - section **Sandboxes** for the selected graph or for all graphs in the selected sandbox. See details in Job config properties (p. 101).

Task "execute_graph" parameters

The "execute graph" task may be triggered by schedule, graph event listener or file event listener. Task editor allows you to specify key=value pairs which are passed to executed graph.

Chapter 19. Manual task execution

Since 3.1

Manual task execution allows you to invoke a task directly with an immediate effect, without defining and triggering an event.

There are a number of task types that are usually associated with a triggering event, such as a file listener or a graph/jobflow listener. You can execute any of these tasks manually.

Additionally, you can specify task parameters to simulate a source event that would normally trigger the task. The following is a figure displaying how a “file event” could be simulated. The parameters for various event sources are listed in the section "Graph parameters".

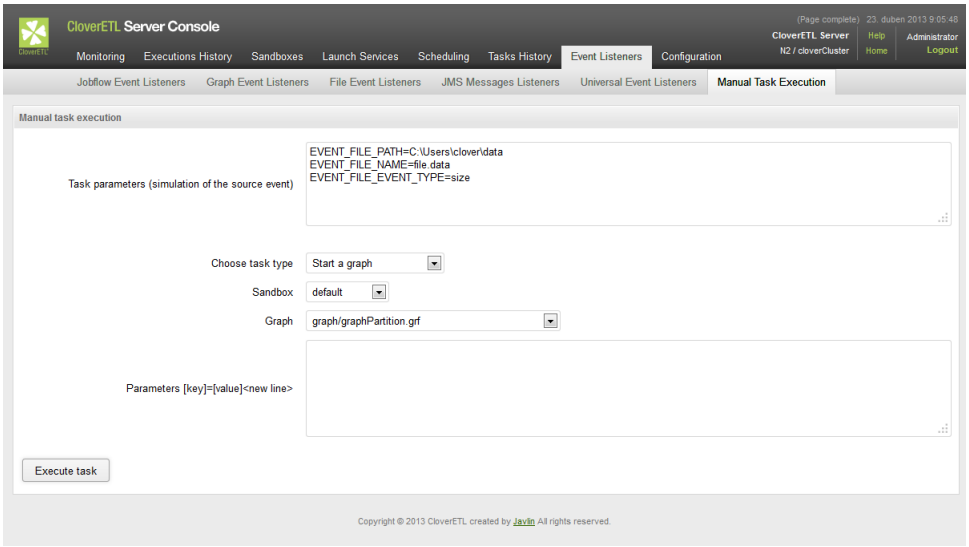


Figure 19.1. Web GUI - "Manual task execution" section

Chapter 20. Scheduling

The scheduling module allows you to create a time schedule for operations you need to trigger in a repetitive or timely manner.

Similar to “cron” from Unix systems, each schedule represents a separate time schedule definition and a task to perform.

In the Cluster, you can explicitly specify which node should execute the scheduled task using the Node ID parameter. However, if not set, the node will be selected automatically from all available nodes (but always just one).

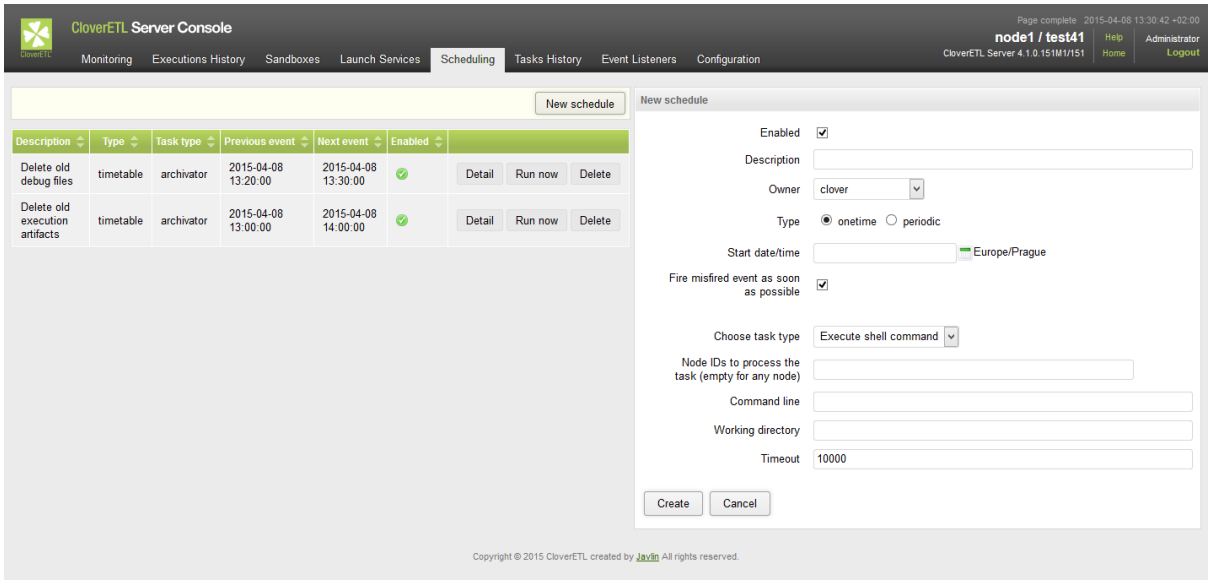


Figure 20.1. Web GUI - section "Scheduling" - create new

Timetable Setting

This section should describe how to specify WHEN schedule should be triggered. Please keep in mind, that exact trigger times are not guaranteed. There may be couple of seconds delay. Schedule itself can be specified in different ways.

- Onetime Schedule (p. 120)
- Periodical schedule by Interval (p. 122)
- Periodical schedule by timetable (Cron Expression) (p. 123)

Onetime Schedule

It is obvious, that this schedule is triggered just once.

Table 20.1. Onetime schedule attributes

Type	"onetime"
Start date/time	Date and time, specified with minutes precision.
Fire misfired ASAP switch	If checked and trigger time is missed because of any reason (i.e. server restart), it will be triggered immediately, when it is possible. Otherwise it is ignored and it will be triggered at next scheduled time.

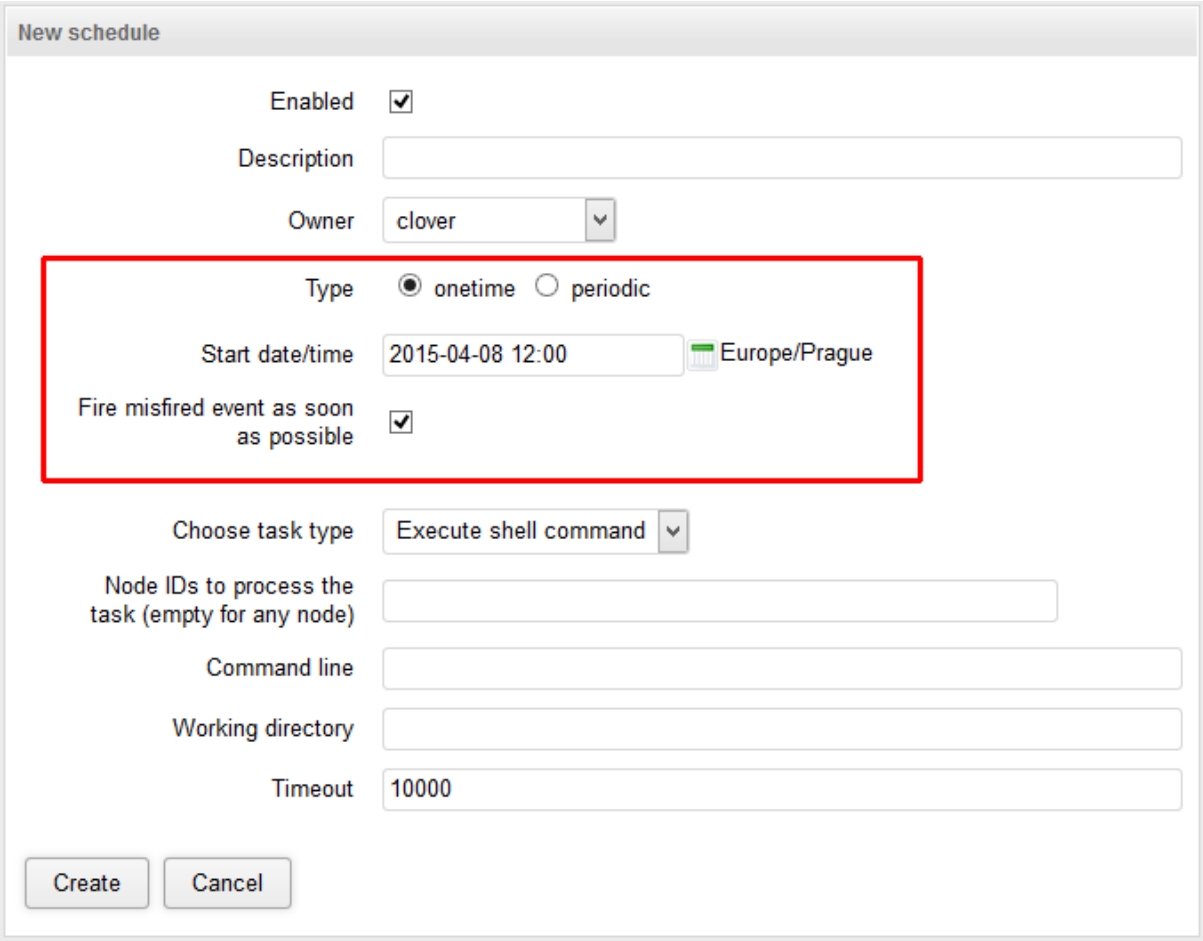


Figure 20.2. Web GUI - onetime schedule form

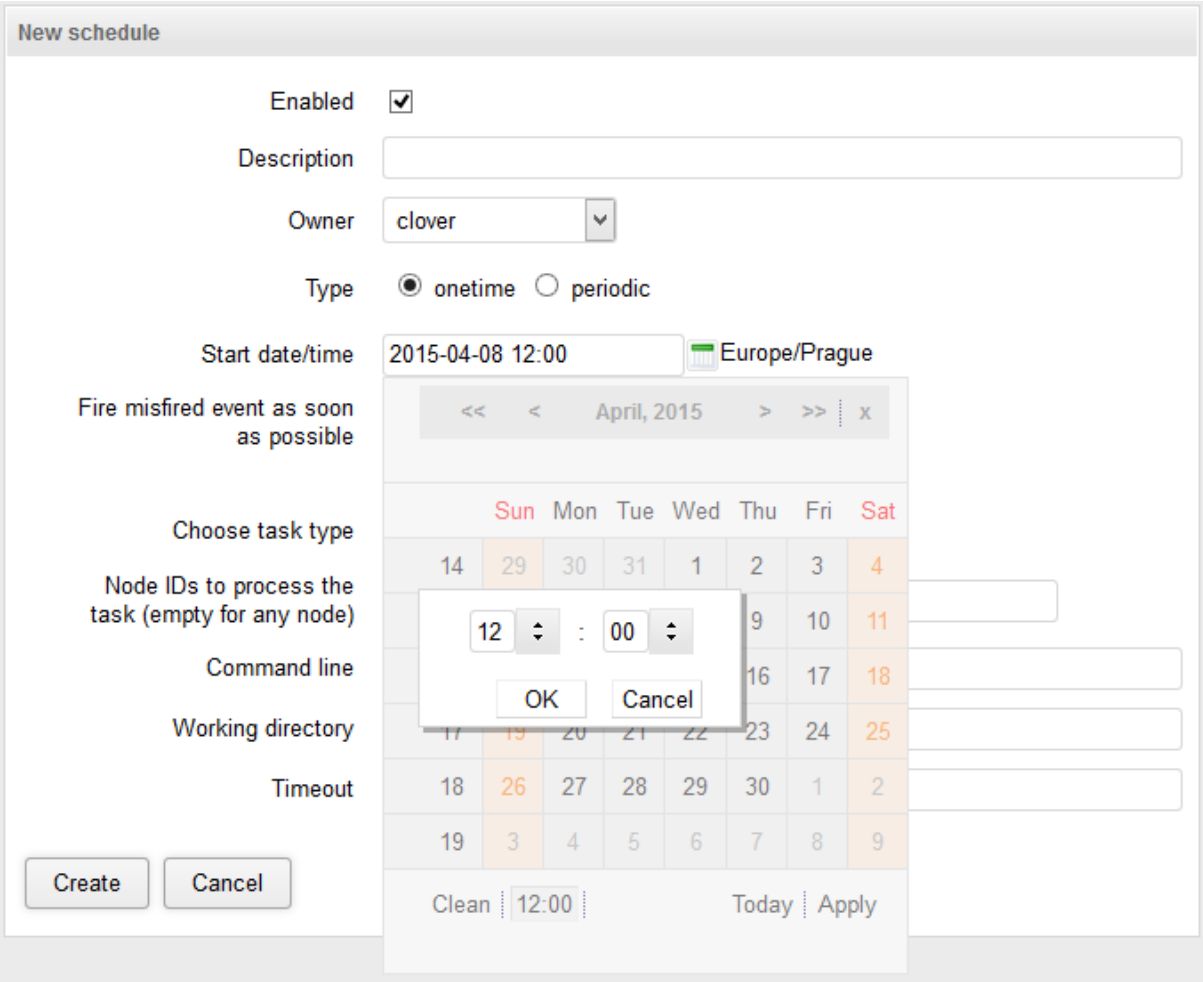


Figure 20.3. Web GUI - schedule form - calendar

Periodical schedule by Interval

This type of schedule is the simplest periodical type. Trigger times are specified by these attributes:

Table 20.2. Periodical schedule attributes

Type	"periodic"
Periodicity	"interval"
Not active before date/time	Date and time, specified with minutes precision.
Not active after date/time	Date and time, specified with minutes precision.
Interval (minutes)	Specifies interval between two trigger times. Next task is triggered even if previous task is still running.
Fire misfired ASAP switch	If checked and trigger time is missed because of any reason (i.e. server restart), it will be triggered immediately, when it is possible. Otherwise it is ignored and it will be triggered at next scheduled time.

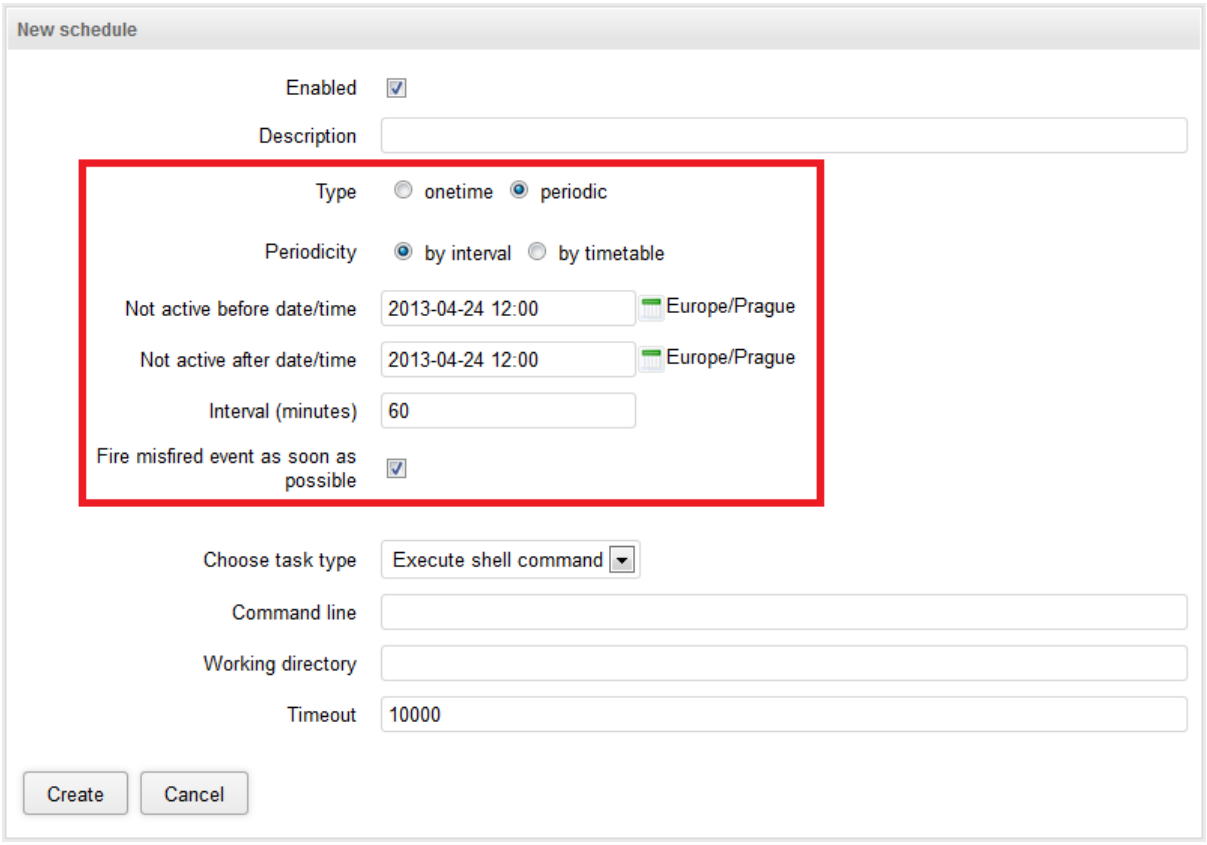


Figure 20.4. Web GUI - periodical schedule form

Periodical schedule by timetable (Cron Expression)

Timetable is specified by powerful (but a little bit tricky) cron expression.

Table 20.3. Cron periodical schedule attributes

Type	"periodic"
Periodicity	"interval"
Not active before date/time	Date and time, specified with minutes precision.
Not active after date/time	Date and time, specified with minutes precision.
Cron expression	Cron is powerful tool, which uses its own format for scheduling. This format is well known among UNIX administrators. i.e. "0 0/2 4-23 * * ?" means "every 2 minutes between 4:00am and 11:59pm".
Fire misfired ASAP switch	If checked and trigger time is missed because of any reason (i.e. server restart), it will be triggered immediately when it is possible. Otherwise it is ignored and it will be triggered at next scheduled time.

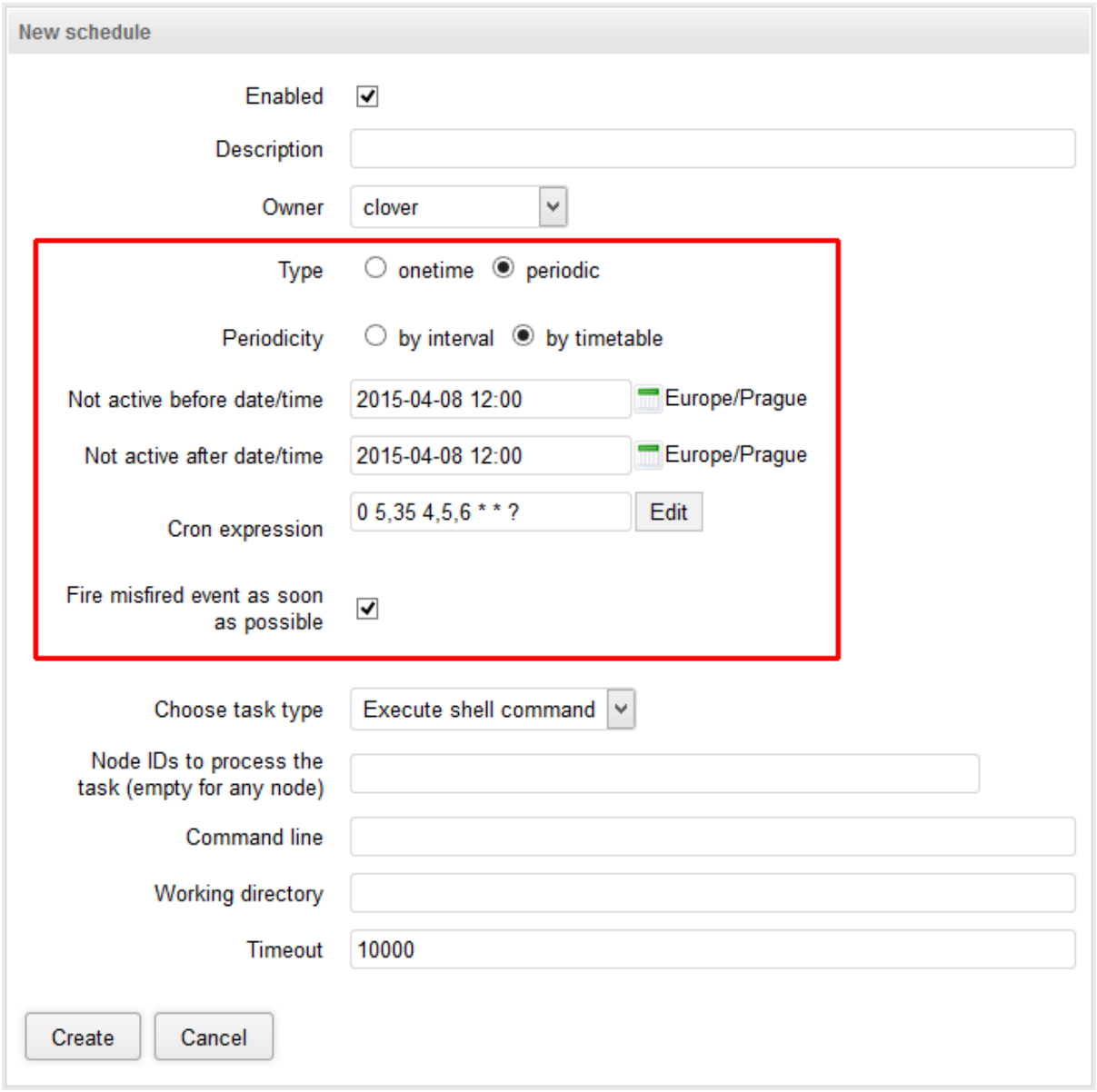


Figure 20.5. Cron periodical schedule form

Tasks

Task basically specifies WHAT to do at trigger time. There are several tasks implemented for schedule and for graph event listener as follows:

- Task - Execution of Graph (p. 125)
- Task - Execution of Jobflow (p. 126)
- Task - Abort job (p. 127)
- Task - Execution of Shell Command (p. 128)
- Task - Send Email (p. 129)
- Task - Execute Groovy Code (p. 129)
- Task - Archivator (p. 130)

In the Cluster environment, all tasks have additional attribute "Node IDs to process the task". It's the comma separated list of cluster nodes, which may process the task. If there is no node ID specified, task may be processed on any cluster node, so in most cases it will be processed on the same node where the event was triggered. If

there are some nodeIDs specified, task will be processed on the first node in the list which is connected in cluster and ready.

Task - Execution of Graph

Please note that behaviour of this task type is almost the same as Task - Execution of Jobflow (p. 126)

Table 20.4. Attributes of "Graph execution" task

Task type	"Start a graph"
Node IDs to process the task	This attribute is accessible only in the cluster environment. It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.
Sandbox	This select box contains sandboxes which are readable for logger user. Select sandbox which contains graph to execute.
Graph	This select box is filled with all graphs files accessible in selected sandbox. Type a graph name or path to filter available items.

Parameters	Key-value pairs which are passed to the executed job as parameters. Besides, if this task is triggered by job (graph or jobflow) event, you can specify source job parameters, which shall be passed from the source job to executed job. i.e. event source has these parameters: paramName2 with value "val2", paramName3 with value "val3", paramName5 with value "val5". Task has "Parameters" attribute set like this:
	
	`paramName1=paramValue1`
	`paramName2=`
	`paramName3`
	`paramName4`
	
	So executed job gets these parameters and values: paramName1 with value "paramValue1" (specified explicitly in the task configuration) paramName2 with value "" (empty string specified explicitly in the task configuration overrides event source parameters), paramName3 with value "val3" (value is taken from event source). These parameters aren't passed: paramName4 isn't passed, since it does not have any value in event source. paramName5 isn't passed, since it is not specified among the parameters to pass in the task.
	
	Event parameters like "EVENT_RUN_RESULT", "EVENT_RUN_ID" etc. are passed to the executed job without limitations.

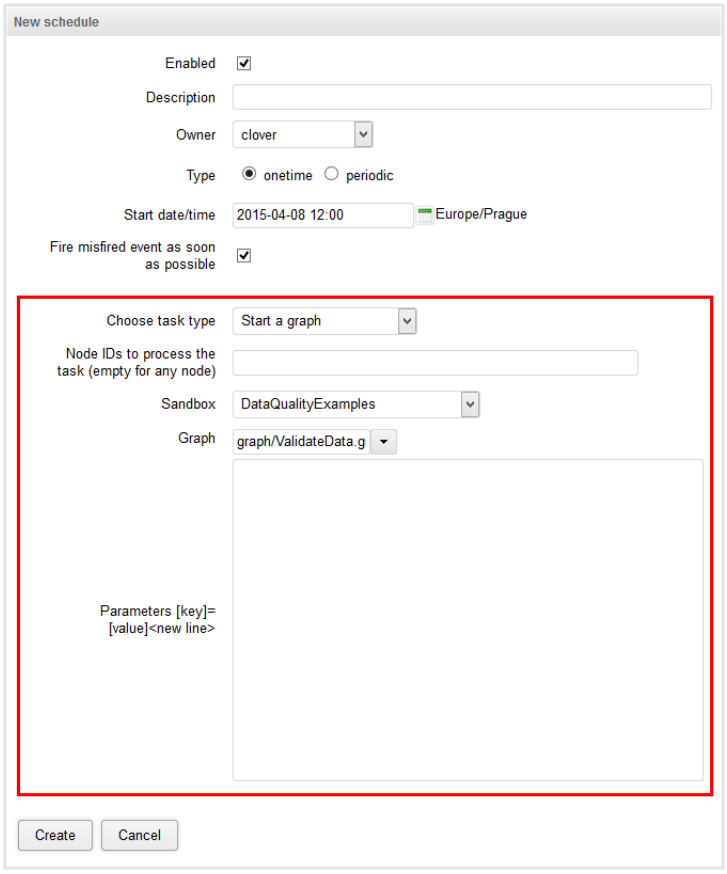


Figure 20.6. Web GUI - Graph execution task

Task - Execution of Jobflow

Please note that behaviour of this task type is almost the same as Task - Execution of Graph (p. 125)

Table 20.5. Attributes of "Jobflow execution" task

Task type	"Start a jobflow"
Node IDs to process the task	This attribute is accessible only in the cluster environment. It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.
Sandbox	This select box contains sandboxes which are readable for logger user. Select sandbox which contains jobflow to execute.
Jobflow	This select box is filled with all jobflow files accessible in selected sandbox. Type jobflow name or path to filter available items.

Parameters	Key-value pairs which are passed to the executed job as parameters. Besides, if this task is triggered by job (graph or jobflow) event, you can specify source job parameters, which shall be passed from the source job to executed job. i.e. event source has these parameters: paramName2 with value "val2", paramName3 with value "val3", paramName5 with value "val5". Task has "Parameters" attribute set like this:
	
	`paramName1=paramValue1`
	`paramName2=`
	`paramName3`
	`paramName4`
	
	So executed job gets these parameters and values: paramName1 with value "paramValue1" (specified explicitly in the task configuration) paramName2 with value "" (empty string specified explicitly in the task configuration overrides event source parameters), paramName3 with value "val3" (value is taken from event source). These parameters aren't passed: paramName4 isn't passed, since it does not have any value in event source. paramName5 isn't passed, since it is not specified among the parameters to pass in the task.
	
	Event parameters like "EVENT_RUN_RESULT", "EVENT_RUN_ID" etc. are passed to the executed job without limitations.

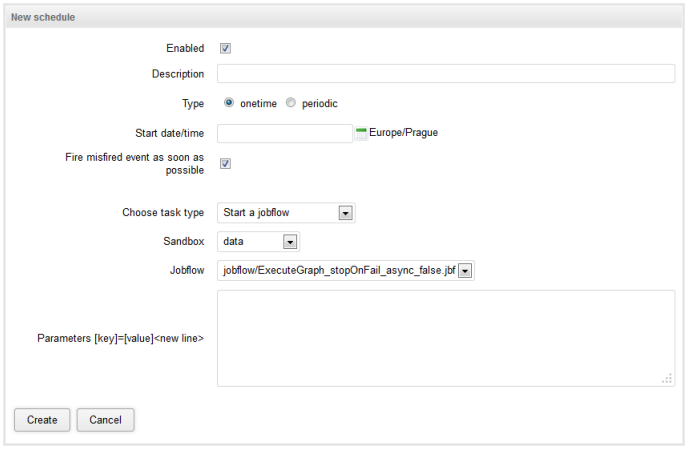


Figure 20.7. Web GUI - Jobflow execution task

Task - Abort job

This task, when activated kills/aborts specified job (ETL graph or jobflow), if it is currently running.

Table 20.6. Attributes of "Abort job" task

Task type	"Abort job"
Node IDs to process the task	This attribute is accessible only in the cluster environment. It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.
Kill source of event	If this switch is on, task will kill job which is source of the event, which activated this task. Attributes sandbox and job are ignored.
Sandbox	Select sandbox which contains job to kill. This attribute works only when "Kill source of event" switch is off.
Job	This select box is filled with all jobs accessible in selected sandbox. All instances of selected job, whose are currently running will be killed. This attribute works only when "Kill source of event" switch is off.

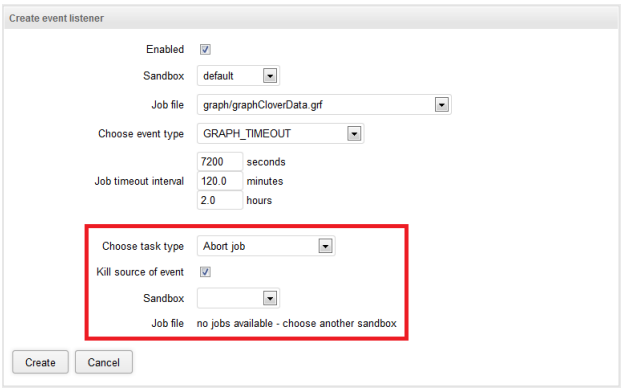


Figure 20.8. Web GUI - "Abort job"

Task - Execution of Shell Command

Table 20.7. Attributes of "Execute shell command" task

Task type	"Execute shell command"
Node IDs to process the task	This attribute is accessible only in the cluster environment. It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.
Command line	Command line for execution of external process.
Working directory	Working directory for process. If not set, working directory of application server process is used.
Timeout	Timeout in milliseconds. After period of time specified by this number, external process is terminated and all results are logged.

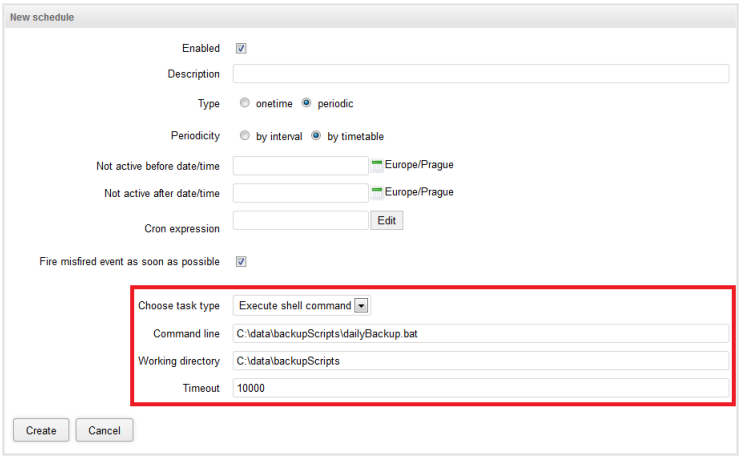


Figure 20.9. Web GUI - shell command

Task - Send Email

This task is very useful, but for now only as response for graph events. This feature is very powerful for monitoring. (see Graph Event Listeners (p. 137) for description of this task type).

Note: It seems useless to send emails periodically, but it may send current server status or daily summary. These features will be implemented in further versions.

Task - Execute Groovy Code

This type of task allows execute code written in script language Groovy. The script can be defined in place or using a path to external `.groovy` file. It is possible to use some variables.

Basic attribute of this task is source code of written in Groovy.

In cluster environment there is also one additional attribute "Node IDs to process the task". It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.

CloverETL Server contains Groovy version 2.0.0

Table 20.8. List of variables available in Groovy code

variable	class	description	availability
event	com.cloveretl.server.events.AbstractServerEvent		every time
task	com.cloveretl.server.persistent.Task		every time
now	java.util.Date	current time	every time
parameters	java.util.Properties	Properties of task	every time
user	com.cloveretl.server.persistent.User	Same as event.getUser()	every time
run	com.cloveretl.server.persistent.RunRecord		When the event is instance of GraphServerEvent
tracking	com.cloveretl.server.persistent.TrackingGraph	same as run.getTrackingGraph()	When the event is instance of GraphServerEvent

variable	class	description	availability
sandbox	com.cloveretl.server.persistent.Sandbox	same as run.getSandbox()	When the event is instance of GraphServerEvent
schedule	com.cloveretl.server.persistent.Schedule	same as ((ScheduleServerEvent)event).getSchedule()	When the event is instance of ScheduleServerEvent
servletContext	javax.servlet.ServletContext		every time
cloverConfiguration	com.cloveretl.server.spring.CloverConfiguration	Configuration values for CloverETL Server	every time
serverFacade	com.cloveretl.server.facade.api.ServerFacade	Reference to the facade interface. Useful for calling CloverETL Server core. WAR file contains JavaDoc of facade API and it is accessible on URL: http://host:port/clover/javadoc/index.html	every time
sessionToken	String	Valid session token of the user who owns the event. It is useful for authorisation to the facade interface.	every time

Variables run, tracking and sandbox are available only if event is instance of GraphServerEvent class. Variable schedule is available only for ScheduleServerEvent as event variable class.

Example of use Groovy script

This example shows script which writes text file describing finished graph. It shows use of 'run' variable.

```
import com.cloveretl.server.persistent.RunRecord;
String dir = "/tmp/";
RunRecord rr = (RunRecord)run;

String fileName = "report"+rr.getId()+"_finished.txt";

FileWriter fw = new FileWriter(new File(dir+fileName));
fw.write("Run ID       :"+rr.getId()+"\n");
fw.write("Graph ID     :"+rr.getGraphId()+"\n");
fw.write("Sandbox      :"+rr.getSandbox().getName()+"\n");
fw.write("\n");
fw.write("Start time   :"+rr.getStartTime()+"\n");
fw.write("Stop time    :"+rr.getStopTime()+"\n");
fw.write("Duration     :"+rr.getDurationString()+"\n");
fw.write("Status     :"+rr.getStatus()+"\n");
fw.close();
```

Task - Archivator

As name suggests, this task can archive (or delete) obsolete records from DB.

Table 20.9. Attributes of "Archivator" task

Task type	"Archivator"
Node IDs to process the task	This attribute is accessible only in the cluster environment. It's comma-separated list of node IDs which may process the task. If it's empty, it may be any node, if there are nodes specified, the task will be processed on the first node which is online and ready.
Older then	Time period (in minutes) - it specifies which records are evaluated as obsolete. Records older then the specified interval are stored in archives.
Archivator type	There are two possible values: "archive" or "delete". Delete removes records without any possibility of UNDO operation. Archive removes records from DB, but creates ZIP package with CSV files containing deleted data.
Output path for archives	This attribute makes sense only for "archive" type.
Include executions history	
Run record with status	If status is selected, only run records with specified status will be archived. It is useful e.g. If you want to delete records for successfully finished jobs, but you want to keep failed jobs for further investigation.
Sandbox	If defined, action is performed for choosen sandbox only. If empty, action is performed for all sandboxes.
Job file	If defined, action is performed for record related to the job file. If empty, action is performed for choosen sandbox (or all sandboxes).
Include temp files	If checked, archivator removes all graph temporary files older then given timestamp defined in "Older than" attribute. The temporary files are files with graph debug data, dictionary files and files created by graph components.
Temp files with record status	If status is selected, only temp files related to the run records with selected status will be archived. It is useful e.g. If you want to delete files for successfully finished jobs, but you want to keep failed jobs for further investigation.
Include tasks history	If checked, archivator will include run records. Log files of graph runs are included as well.
Task types	If this task type is selected, only logs for selected task type are archived.
Task result mask	Mask applied to task log result attribute. Only records whose result meets this mask are archived. Specify string without any wildcards. Each task log which contains specified string in the "result" attribute will be deleted/ archived. Case sensitivity depends on database collation.
Include profiler runs	If checked, archivator will include profiler job results.

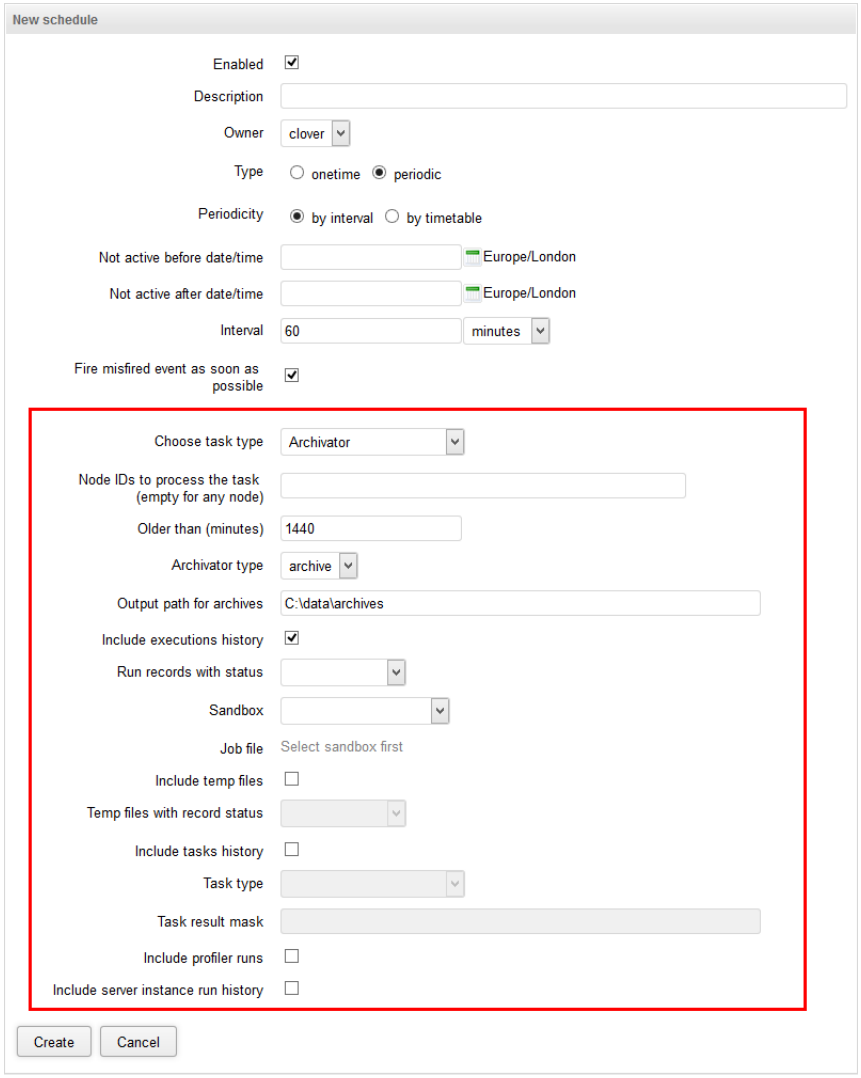


Figure 20.10. Web GUI - archive records

Chapter 21. Viewing Job Runs - Executions History

Executions History shows the history of all jobs that the Server has executed – transformation graphs, jobflows, and Data Profiler jobs. You can use it to find out why a job failed, see the parameters that were used for a specific run, and much more.

The table shows basic information about the job: Run ID, Job file, current status, and time of execution, as well as some useful links. You will also find additional details after clicking on the job name in the list – details such as associated log files, parameter values, tracking, and more.

Please note that some jobs might not appear in the Executions History list. These are jobs that have disabled persistency for increased performance – e.g. some Launch Services disable storing the run information in order to increase service responsiveness.

Filtering and ordering

Use the Filter panel to filter the view. By default, only parent tasks are shown (Show executions children) – e.g. master nodes in the Cluster and their workers are hidden by default.

Use the up and down arrows in the table header to sort the list. By default, the latest job is listed first.

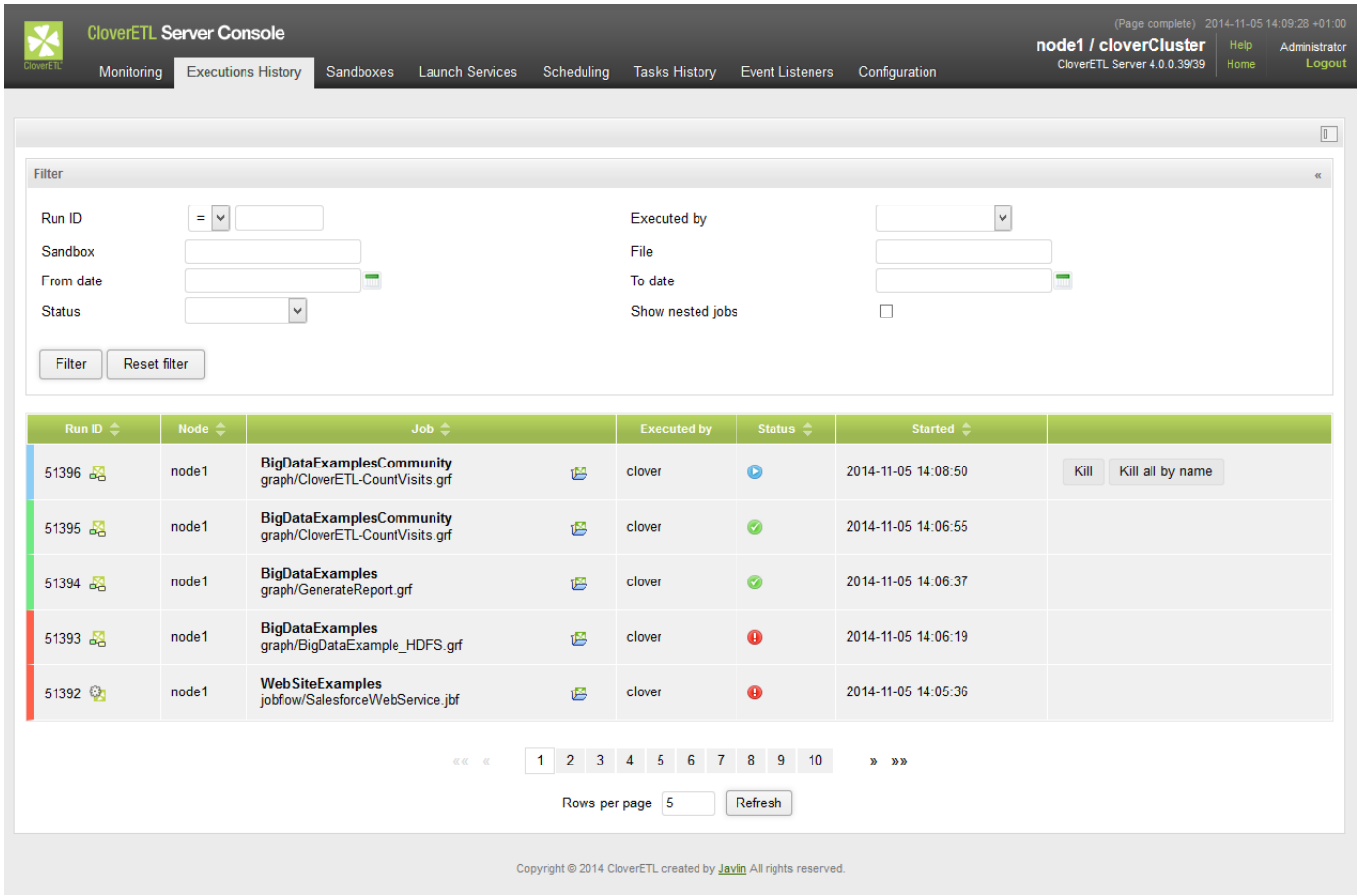


Figure 21.1. Executions History - executions table

When some job execution is selected in the table, the detail info is shown on the right side.

Table 21.1. Persistent run record attributes

Attribute	Description
Run ID	Unique number identifying the run of the job. Server APIs usually return this number as simple response of execution request. It's useful as parameter of subsequent calls for specification of the job execution.
Execution type	Type of job as recognized by the server. STANDALONE for ETL graph, JOBFLOW for Jobflow, PROFILER_JOB for profiler, MASTER for main record of partitioned execution in cluster, PARTITIONED_WORKER for worker record of partitioned execution in cluster
Parent run ID	Run ID of the parent job. Typically Jobflow which executed this job, or master execution which encapsulate this worker execution.
Root run ID	Run ID of the root parent job. Job execution which wasn't executed by another parent job.
Execution group	Jobflow components may group sub-jobs using this attribute. See description of Jobflow componentes for details.
Nested jobs	Indication that this job execution has or has not any child execution.
Node	In cluster mode shows ID of the cluster node which this execution was running on.
Executed by	User which executed the job. Either directly using some API/GUI or indirectly using the scheduling or event listeners.
Sandbox	Sandbox containing job file. For jobs which are sent together with execution request, so the job file doesn't exist on the server site, it's set to "default" sandbox.
Job file	Path to job file, relative to the sandbox root. For jobs which are sent together with execution request, so the job file doesn't exist on the server site, it's set to generated string.
Job version	Revision of the job file. It's string generated by CloverETL Designer and stored in the job file.
Status	Status of the job execution. READY - waiting for execution start, RUNNING - processing job, FINISHED OK - job finished without any error, ABORTED - job was aborted directly using some API/GUI or by parent Jobflow, ERROR - job failed, N/A (not awailable) - server process died suddenly, so it couldn't properly abort the jobs, so after restart the jobs with unknown status are set as N/A
Started	Server date-time (and timezone) of the execution start.
Finished	Server date-time (and timezone) of the execution finish.
Duration	Execution duration
Error in component ID	If the job failed due the error in a component, this field contains ID of the component.
Error in component type	If the job failed due the error in a component, this field contains type of the component.
Error message	If the job failed, this field contains error description.
Exception	If the job failed, this field contains error stack trace.
Input parameters	List of input parameters passed to the job. Job file can't be cached, since the parameters are applied during loading from the job file. Job file isn't cached by default.
Input dictionary	List of dictionary elements passed to the job. Dictionary is used independently on job file caching.
Output dictionary	List of dictionary elements at the moment the job ends.

For jobs which have some children executions, e.g. partitioned or jobflows also executions hierarchy tree is shown.

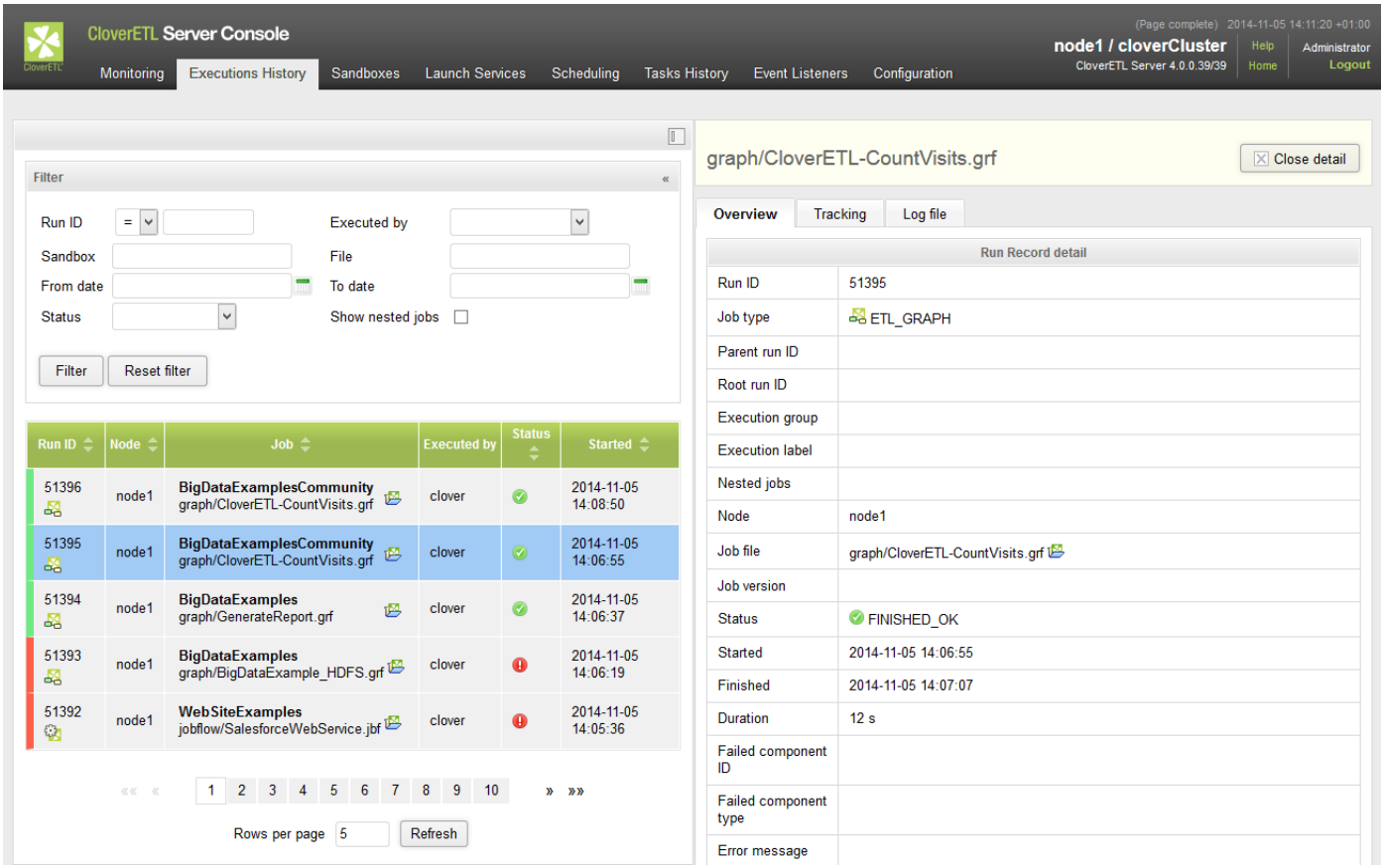


Figure 21.2. Executions History - overall perspective

Since the detail panel and expecially job logs may be wide, it may be usefull to hide table on the left, so the detail panel spreads. Click on the minimize icon on the top of the list panel to hide panel. Then to show list panel agan, click to the "Expand panel" icon on the left.

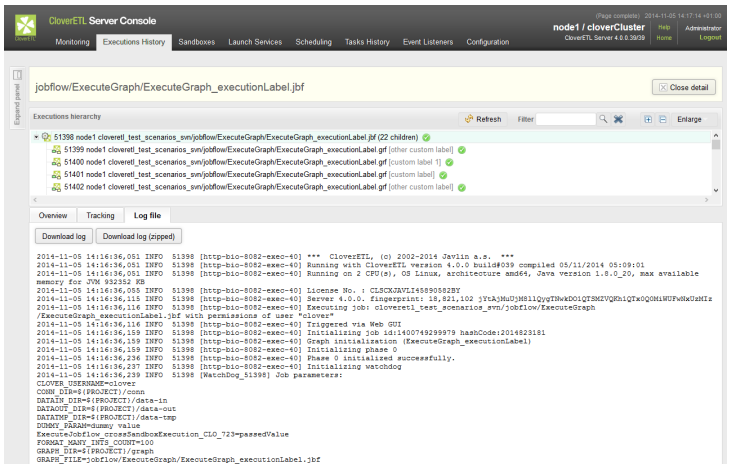


Figure 21.3. Executions Hierarchy with docked list of jobs

Executions hierarchy may be rather complex, so it's possible to filter the content of the tree by fulltext filter. However when the filter is used, the selected executions aren't hierarchically structured.

Chapter 22. Listeners

Listeners can be seen as hooks. They wait for a specific event and take a used-defined action if event occurs.

The event are specific to the particular listener (Graph Event Listeners (p. 137), Jobflow Event Listeners (p. 145), JMS messages listeners (p. 147), Universal event listeners (p. 151) or File event listeners (p. 152)). The available actions taken by the listeners are common for all listeners.

The action, that can be taken are:

- Send email - see Task - Send Email (p. 139)
- Execute shell command - see Task - Execution of Shell Command (p. 128)
- Start a graph - see Task - Execution of Graph (p. 125)
- Start a jobflow - see Task - Execution of Jobflow (p. 126)
- Start a profiler job
- Abort job - see Task - Abort job (p. 127)
- Archivator - see Task - Archivator (p. 130)
- Send a JMS Message - see Task - JMS Message (p. 141)
- Execute Groovy code - see Task - Execute Groovy Code (p. 129)

Graph Event Listeners

Graph Event Listeners allow you to define a task that the Server will execute as a reaction to the success, failure or other event of a specific job (a transformation graph).

Each listener is bound to a specific graph and is evaluated no matter whether the graph was executed manually, scheduled, or via an API call, etc.

You can use listeners to chain multiple jobs (creating a success listener that starts the next job in a row). However, we recommend using Jobflows to automate complex processes because of its better development and monitoring capabilities.

Graph Event Listeners are similar to Jobflow Event Listeners (Jobflow Event Listeners (p. 145)) – for CloverETL Server both are simply “jobs”.

In the Cluster, the event and the associated task are executed on the same node the job was executed on by default. If the graph is distributed, the task will be executed on the master worker node. However, you can override where the task will be executed by explicitly specifying a Node IDs in the task definition.

Graph Events

Each event carries properties of graph, which is source of event. If there is an event listener specified, task may use these properties. i.e. next graphs in the chain may use "EVENT_FILE_NAME" placeholder which activated first graph in the chain. Graph properties, which are set specifically for each graph run (i.e. RUN_ID), are overridden by last graph.

For now, there are these types of graph events:

graph started

Event of this type is created, when ETL graph execution successfully started.

graph phase finished

Event of this type is created, everytime when graph phase is finished and all its nodes are finished with status `FINISHED_OK`.

graph finished OK

Event of this type is created, when all phases and nodes of graph are finished with status `FINISHED_OK`.

graph error

Event of this type is created, when graph cannot be executed from any reason, or when any node of graph fails.

graph aborted

Event of this type is created, when graph is explicitly aborted.

graph timeout

Event of this type is created, when graph runs longer then specified interval. Thus you have to specify "Job timeout interval" attribute for each listener of graph timeout event. You can specify this interval in seconds or in minutes or in hours.

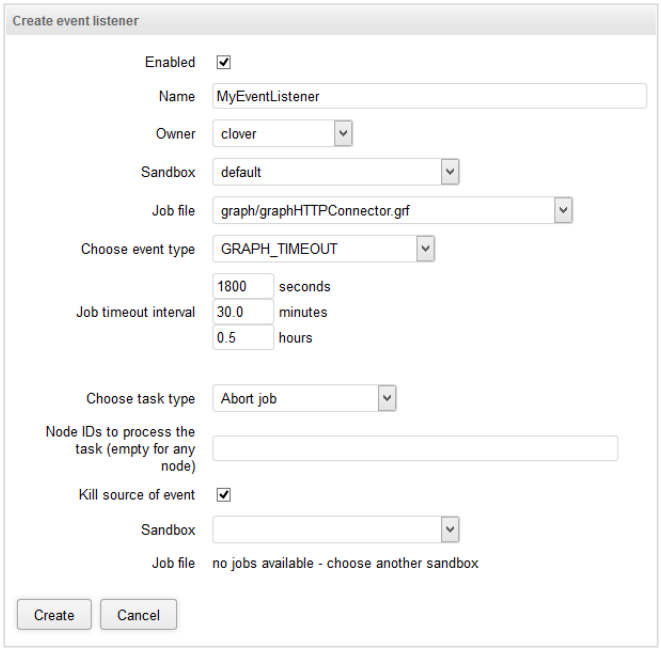


Figure 22.1. Web GUI - graph timeout event

graph status unknown

Event of this type is created, when the server, during the startup, detects run records with undefined status in the executions history. Undefined status means, that server has been killed during graph run. Server automatically changes state of graph to "Not Available" and sends 'graph status unknown' event. Please note, that this works just for executions, which have persistent record in executions history. It is possible to execute transformation without persistent record in executions history, typically for better performance of fast running transformations (i.e. using Launch Services).

Listener

User may create listener for specified event type and graph (or all graphs in sandbox). Listener is actually connection between graph event and task, where graph event specifies WHEN and task specifies WHAT to do.

So progress is like this:

- event is created
- listeners for this event are notified
- each listener performs related task

Tasks

Task types "execute shell command", "execute graph" and "archivator" are described in Chapter 20, Scheduling (p. 120), see this chapter for details about these task types. There is one more task type, which is useful especially with graph event listeners, thus it is described here. It is task type "send email".

Note: You can use task of any type for both scheduling and graph event listener. Description of task types is divided into two sections just to show the most obvious use cases.

In the Cluster environment, all tasks have additional attribute "Node IDs to process the task". It's the comma separated list of cluster nodes, which may process the task. If there is no node ID specified, task may be processed on any cluster node, so in most cases it will be processed on the same node where the event was triggered. If there are some nodeIDs specified, task will be processed on the first node in the list which is connected in cluster and ready.

Task - Send Email

This type of task is useful for notifications about result of graph execution. I.e. you can create listener with this task type to be notified about each failure in specified sandbox or failure of particular graph.

Table 22.1. Attributes of "Send email" task

Task type	"email"
E-mail template	This select box is available only when user is creating new record. It contains all predefined email templates. If user chooses any of them, all fields below are automatically filled with values from template.
To	Recipient's email address. It is possible to specify more addresses separated by comma. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
Cc	Cc stands for 'carbon copy'. Copy of email will be delivered to these addresses. It is possible to specify more addresses separated by comma. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
Bcc	Bcc: stands for 'Blind carbon copy'. It is the same as Cc, but the others recipients aren't aware, that these recipients get copy of email.
Reply-to (Sender)	Email address of sender. It must be valid address according to SMTP server. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
Subject	Email subject. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
Text	Body of email in plain text. Email is created as multipart, so HTML body should have a precedence. Plain text body is only for email clients which do not display HTML. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
HTML	Body of email in HTML. Email is created as multipart, so HTML body should have a precedence. Plain text body is only for email clients which do not display HTML. It is also possible to use placeholders. *See* Placeholders *(p. 140) for details.*
Log file as attachment	If this switch is checked, email will have an attachment with packed log file of related graph execution.

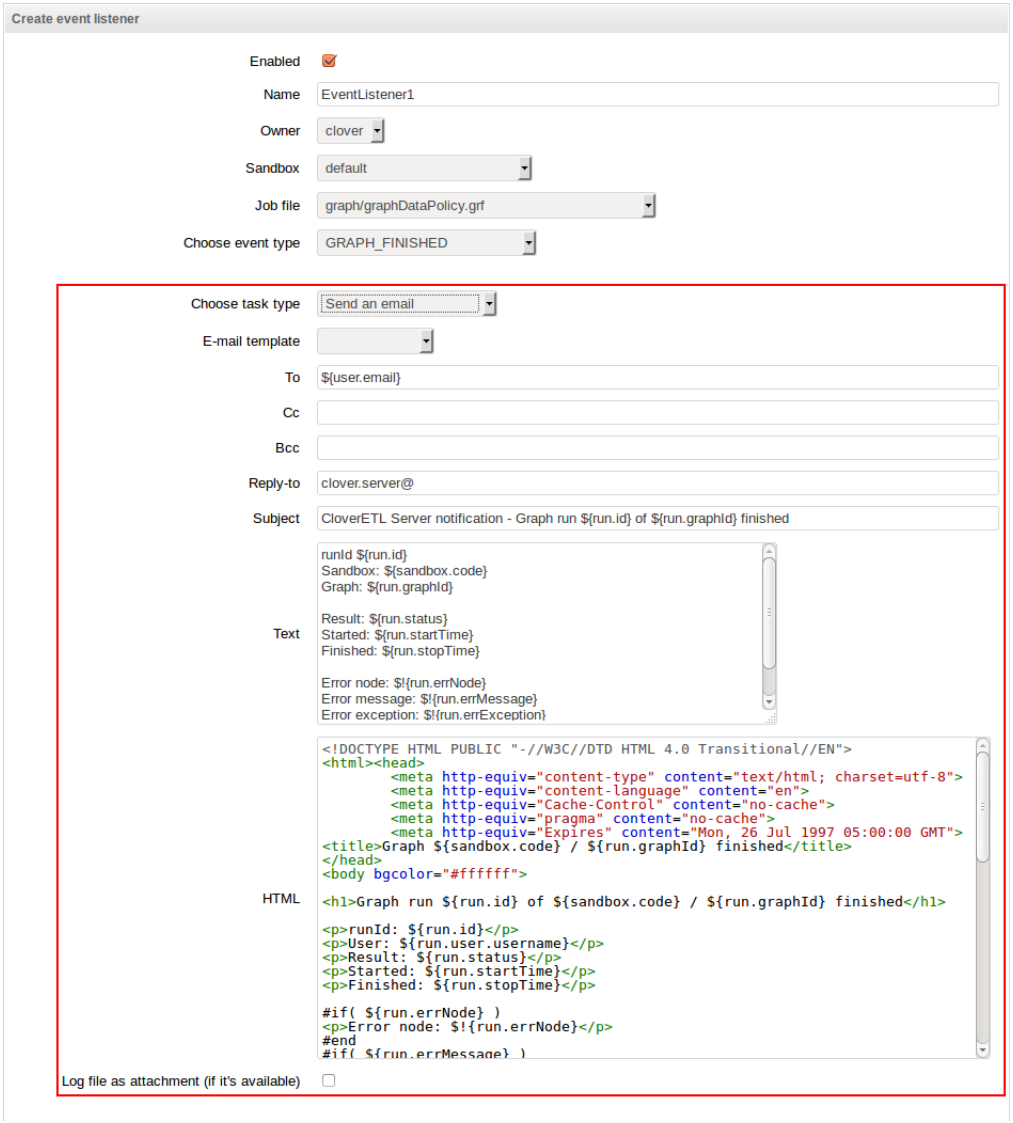


Figure 22.2. Web GUI - send email

Note: Do not forget to configure connection to SMTP server (See Part III, "Configuration" (p. 40) for details).

Placeholders

Placeholder may be used in some fields of tasks. They are especially useful for email tasks, where you can generate content of email according to context variables.

Note: In most cases, you can avoid this by using email templates (See Email task for details)

These fields are preprocessed by Apache Velocity templating engine. See Velocity project URL for syntax description http://velocity.apache.org/

There are several context variables, which you can use in placeholders and even for creating loops and conditions.

- *event*
- *now*
- *user*
- *run*
- *sandbox*

Some of them may be empty in dependence of occasion which field is processed in. I.e. If task is processed because of graph event, then *run* and *sandbox* variables contain related data, otherwise they are empty,

Table 22.2. Placeholders useful in email templates

Variable name	Contains
now	Current date-time
user	User, who caused this event. It may be owner of schedule, or someone who executed graph. Contains sub-properties, which are accessible using dot notation (i.e. ${user.email}) email, username, firstName, lastName, groups (list of values)
run	Data structure describing one single graph execution. Contains sub-properties, which are accessible using dot notation (i.e. ${run.jobFile}) jobFile, status, startTime, stopTime, errNode, errMessage, errException, logLocation

tracking	Data structure describing status of components in graph execution. Contains sub-properties, which are accessible using Velocity syntax for loops and conditions.
	`#if (${tracking})`
	`<table border="1" cellpadding="2" cellspacing="0">`
	`#foreach ($phase in $tracking.trackingPhases)`
	`<tr><td>phase: ${phase.phaseNum}</td>`
	`<td>${phase.executionTime} ms</td>`
	`<td></td><td></td><td></td><td></tr>`
	`#foreach ($node in $phase.trackingNodes)`
	`<tr><td>${node.nodeName}</td>`
	`<td>${node.result}</td>`
	`<td></td><td></td><td></td><td></tr>`
	`#foreach ($port in $node.trackingPorts)`
	`<tr><td></td><td></td>`
	`<td>${port.type}:${port.index}</td>`
	`<td>${port.totalBytes} B</td>`
	`<td>${port.totalRows} rows</td></tr>`
	`#end`
	`#end`
	`#end`
	`</table>`
	`#end`
	`}`

sandbox	Data structure describing sandbox containing executed graph. Contains sub-properties, which are accessible using dot notation (i.e. ${sandbox.name}) name, code, rootPath
schedule	Data structure describing schedule which triggered this task. Contains sub-properties, which are accessible using dot notation (i.e. ${schedule.description}) description, startTime, endTime, lastEvent, nextEvent, fireMisfired

Task - JMS Message

This type of task is useful for notifications about result of graph execution. I.e. you can create graph event listener with this task type to be notified about each failure in specified sandbox or failure of particular graph.

JMS messaging requires JMS API (jms.jar) and third-party libraries. All these libraries must be available on application server classpath. Some application servers contain these libraries by default, some do not, thus the libraries must be added explicitly.

Table 22.3. Attributes of JMS message task

Task type	"JMS message"
Initial context class name	Full class name of javax.naming.InitialContext implementation. Each JMS provider has own implementation. i.e. for Apache MQ it is "org.apache.activemq.jndi.ActiveMQInitialContextFactory". If it is empty, server uses default initial context
Connection factory JNDI name	JNDI name of connection factory. Depends on JMS provider.
Destination JNDI name	JNDI name of message queue/topic on the server
Username	Username for connection to JMS message broker
Password	Password for connection to JMS message broker
URL	URL of JMS message broker
JMS pattern	This select box is available only when user is creating new record. It contains all predefined JMS message patterns. If user chooses any of them, text field below is automatically filled with value from pattern.
Text	Body of JMS message. It is also possible to use placeholders. *See* Placeholders *(p. 140) of send email task for details.*

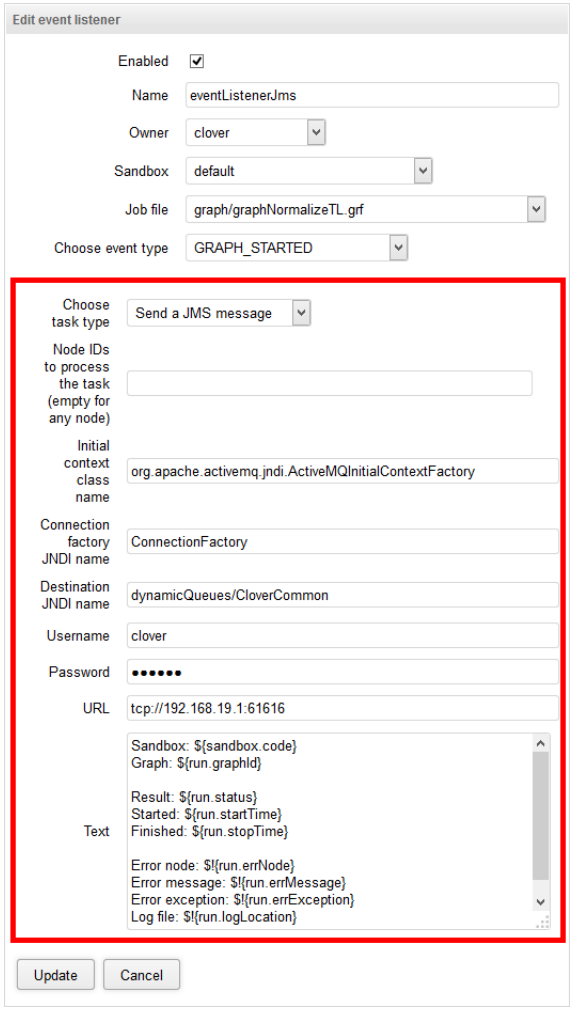


Figure 22.3. Web GUI - Task JMS message editor

Use cases

Possible use cases are the following:

Execute graphs in chain

Let's say, that we have to execute graph B, only if another graph A finished without any error. So there is some kind of relation between these graphs. We can achieve this behaviour by creating graph event listener. We create listener for event `graph finished OK` of graph A and choose task type `execute graph` with graph B specified for execution. And that is it. If we create another listener for graph B with task `execute graph` with graph C specified, it will work as chain of graphs.

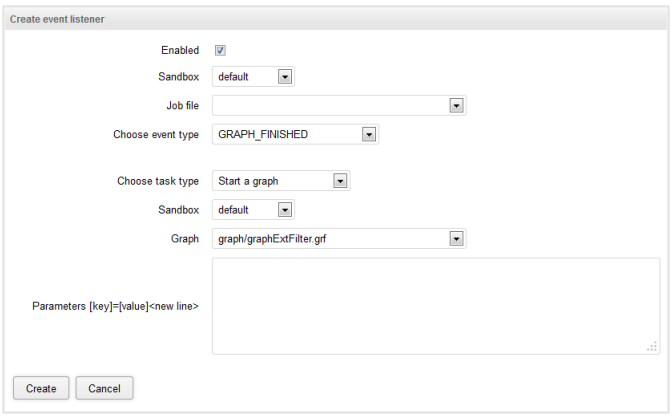


Figure 22.4. Event source graph isn't specified, thus listener works for all graphs in specified sandbox

Email notification about graph failure

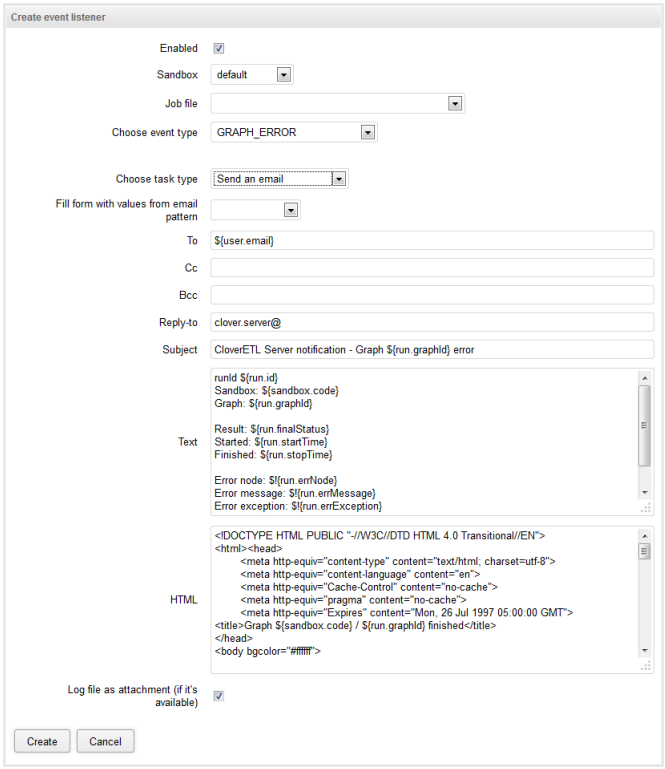


Figure 22.5. Web GUI - email notification about graph failure

Email notification about graph success

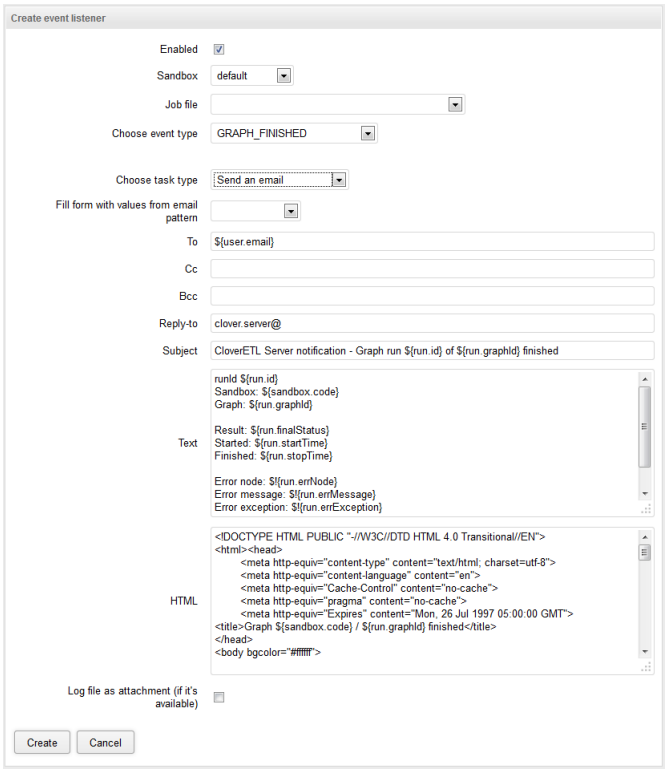


Figure 22.6. Web GUI - email notification about graph success

Backup of data processed by graph

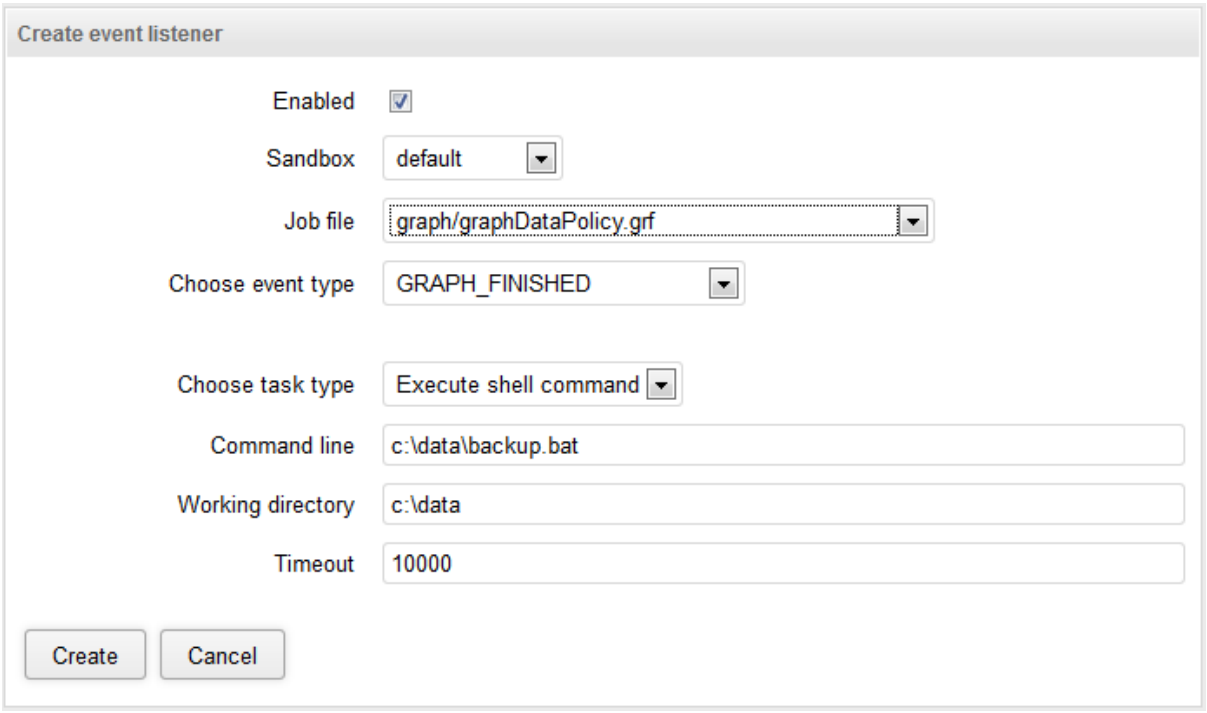


Figure 22.7. Web GUI - backup of data processed by graph

Jobflow Event Listeners

Jobflow Event Listeners allow you to define a task that the Server will execute as a reaction to the success or failure of executing a specific job (a jobflow).

Each listener is bound to a specific jobflow and is evaluated every time the jobflow is executed (no matter whether manually, through another jobflow, scheduled, via an API call, etc.).

Jobflow Event Listeners work very similarly to Graph Event Listeners (Tasks (p. 139)) in many ways, since ETL Graphs and Jobflows are both "jobs" from the point of view of the CloverETL Server.

In the Cluster, the event and the associated task are executed on the same node the job was executed on. If the jobflow is distributed, the task will be executed on the master worker node. However, you can override where the task will be executed by explicitly specifying a Node ID in the task definition.

Jobflow Events

Each event carries properties of the event source job. If there is an event listener specified, task may use these properties. e.g. next job in the chain may use "EVENT_FILE_NAME" placeholder which activated first job in the chain. Job properties, which are set specifically for each run (e.g. RUN_ID), are overridden by last job.

There are these types of jobflow events:

- jobflow started (p. 146)
- jobflow phase finished (p. 146)
- jobflow finished OK (p. 146)
- jobflow error (p. 146)
- jobflow aborted (p. 146)
- jobflow timeout (p. 146)

- jobflow status unknown (p. 146)

jobflow started

Event of this type is created, when jobflow execution successfully started.

jobflow phase finished

Event of this type is created, everytime when jobflow phase is finished and all its nodes are finished with status `FINISHED_OK`.

jobflow finished OK

Event of this type is created, when all phases and nodes of jobflow are finished with status `FINISHED_OK`.

jobflow error

Event of this type is created, when jobflow cannot be executed from any reason, or when any node of the jobflow fails.

jobflow aborted

Event of this type is created, when jobflow is explicitly aborted.

jobflow timeout

Event of this type is created, when jobflow runs longer then specified interval. Thus you have to specify "Job timeout interval" attribute for each listener of jobflow timeout event. You can specify this interval in seconds or in minutes or in hours.

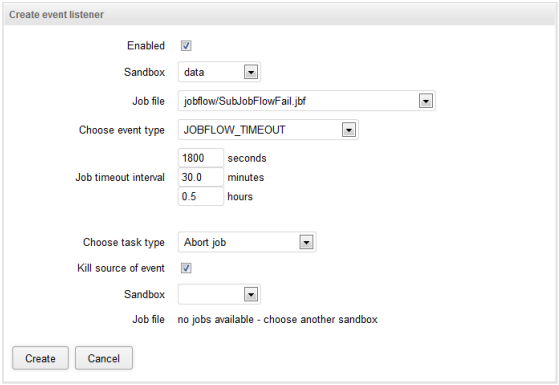


Figure 22.8. Web GUI - jobflow timeout event

jobflow status unknown

Event of this type is created, when the server, during the startup, detects run records with undefined status in the executions history. Undefined status means, that server has been killed during jobflow run. Server automatically changes state of jobflow to "Not Available" and sends 'jobflow status unknown' event. Please note, that this works just for executions, which have persistent record in executions history. It is possible to execute transformation without persistent record in executions history, typically for better performance of fast running transformations (e.g. using Launch Services).

Listener

User may create listener for specified event type and jobflow (or all jobflows in sandbox). Listener is actually connection between jobflow event and task, where jobflow event specifies WHEN and task specifies WHAT to do.

So progress is like this:

- event is created
- listeners for this event are notified
- each listener performs related task

Tasks

Task specifies operation which should be performed as the reaction to the triggered event.

Task types are described in Tasks (p. 124) and Tasks (p. 139)

Note: You can use task of any type for jobflow event listener. Description of task types is divided into two sections just to show the most obvious use cases.

JMS messages listeners

JMS Message Listeners allow you to listen for incoming JMS messages. You specify the source of the messages (JMS Topic or JMS Queue) and a task that will be executed for each incoming message.

JMS messaging requires a JMS API (jms.jar) and specific third-party libraries. Every one of these libraries must be available on an application server classpath. Some application servers contain these libraries by default; however, some do not. In such a case, libraries must be added explicitly before starting the CloverETL Server.

JMS is a complex topic that goes beyond the scope of this document. For more detailed information about JMS, refer to the Oracle website: http://docs.oracle.com/javaee/6/tutorial/doc/bncdq.html

Note that the JMS implementation is dependent on the application server that the CloverETL Server is running in.

In Cluster, you can either explicitly specify which node will listen to JMS or not. If unspecified, all nodes will register as listeners. In the case of JMS Topic, all nodes will get the message and will trigger the task (multiple instances) or, in the case of JMS Queue, a random node will consume the message and will run the task (just one instance).

Table 22.4. Attributes of JMS message task

Attribute	Description
Node ID to handle the event	This attribute makes sense only in cluster environment. It is node ID where the listener should be initialized. If it is not set, listener is initialized on all nodes in the cluster. In the Cluster environment, each JMS event listener has a "Node IDs" attribute which may be used for specification which cluster node will consume messages from the queue/topic. There are following possibilities: • No failover: Just one node ID specified - Only specified node may consume messages, however node status must be "ready". When the node isn't ready, messages aren't consumed by any cluster node. • Failover with node concurrency: No node ID specified (empty input) - All cluster nodes with status "ready" concurently consume messages.

Attribute	Description
	• Failover with node reservation: More node IDs specified (separated by comma) - Just one of specified nodes consumes messages at a time. If the node fails from any reason (or its status isn't "ready"), any other "ready" node from the list continues with consuming messages. In standalone environment, the "Node IDs" attribute is ignored.
Initial context class name	Full class name of javax.naming.InitialContext implementation. Each JMS provider has own implementation. i.e. for Apache MQ it is "org.apache.activemq.jndi.ActiveMQInitialContextFactory". If it is empty, server uses default initial context. Specified class must be on web-app classpath or application-server classpath. It is usually included in one library with JMS API implementation for each specific JMS broker provider.
Connection factory JNDI name	JNDI name of connection factory. Depends on JMS provider.
Destination JNDI name	JNDI name of message queue/topic on the server
Username	Username for connection to JMS message broker
Password	Password for connection to JMS message broker
URL	URL of JMS message broker
Durable subscriber (only for Topics)	If it is false, message consumer is connected to the broker as "non-durable", so it receives only messages which are sent while the connection is active. Other messages are lost. If it is true, consumer is subscribed as "durable" so it receives even messages which are sent while the connection is inactive. The broker stores such messages until they can be delivered or until the expiration is reached. This switch makes sense only for Topics destinations, because Queue destinations always store messages until they can be delivered or the expiration is reached. Please note, that consumer is inactive i.e. during server restart and during short moment when user updates the "JMS message listener" ant it must be re-initialized. So during these intervals the message in the Topic may get lost if the consumer does not have durable subscription. If the subscription is durable, client must have "ClientId" specified. This attribute can be set in different ways in dependence of JMS provider. I.e. for ActiveMQ, it is set as URL parameter tcp://localhost:1244?jms.clientID=TestClientID
Message selector	This "query string" can be used as specification of conditions for filtering incoming messages. Syntax is well described on Java EE API web site: http://java.sun.com/j2ee/1.4/docs/api/javax/jms/Message.html It has different behaviour depending on type of consumer (queue/topic) Queue: If a its a queue the messages that are filtered out remain on the queue. Topic: Messages filtered out by a Topic subscriber's message selector will never be delivered to the subscriber. From the subscriber's perspective, they do not exist.
Groovy code	Groovy code may be used for additional message processing and/or for refusing message. Both features are described below.

Optional Groovy code

Groovy code may be used for additional message processing or for refusing a message.

- **Additional message processing** Groovy code may modify/add/remove values stored in the containers "properties" and "data".

- **Refuse/acknowledge the message** If the Groovy code returns Boolean.FALSE, the message is refused. Otherwise, the message is acknowledged. A refused message may be redelivered, however the JMS broker

should configure a limit for redelivering messages. If the groovy code throws an exception, it’s considered a coding error and the JMS message is NOT refused because of it. So, if the message refusal is to be directed by some exception, it must be handled in groovy.

Table 22.5. Variables accessible in groovy code

type	key	description
javax.jms.Message	msg	instance of JMS message
java.util.Properties	properties	See below for details. Contains values (String or converted to String) read from message and it is passed to the task which may use them somehow. I.e. task "execute graph" passes these parameters to the executed graph.
java.util.Map<String, Object>	data	See below for details. Contains values (Object, Stream, ..) read or proxied from the message instance and it is passed to the task which may use them somehow. I.e. task "execute graph" passes it to the executed graph as "dictionary entries".
javax.servlet.ServletContext	servletContext	instance of ServletContext
javax.jms.Message	msg	instance of JMS message
com.cloveretl.server.api.ServerFacade	serverFacade	instance of serverFacade usable for calling CloverETL Server core features.
String	sessionToken	sessionToken, needed for calling serverFacade methods

Message data available for further processing

A JMS message is processed and the data it contains is stored in two data structures: Properties and Data.

Table 22.6. Properties Elements

key	description
JMS_PROP_[property key]	For each message property is created one entry, where "key" is made of prefix "JMS_PROP_" and property key.
JMS_MAP_[map entry key]	If the message is instance of MapMessage, for each map entry is created one entry, where "key" is made of prefix "JMS_MAP_" and map entry key. Values are converted to String.
JMS_TEXT	If the message is instanceof TextMessage, this property contains content of the message.
JMS_MSG_CLASS	Class name of message implementation
JMS_MSG_CORRELATIONID	Correlation ID is either provider-specific message ID or application-specific String value
JMS_MSG_DESTINATION	The JMSDestination header field contains the destination to which the message is being sent.
JMS_MSG_MESSAGEID	A JMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.
JMS_MSG_REPLYTO	Destination to which a reply to this message should be sent.
JMS_MSG_TYPE	Message type identifier supplied by the client when the message was sent.
JMS_MSG_DELIVERYMODE	The DeliveryMode value specified for this message.
JMS_MSG_EXPIRATION	The time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send.
JMS_MSG_PRIORITY	The JMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.
JMS_MSG_REDELIVERED	"true" if this message is being redelivered.
JMS_MSG_TIMESTAMP	The time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queueing of messages.

Note that all values in the “Properties” structure are stored as String type – however they are numbers or text.

For backwards compatibility, all listed properties can also be accessed using lower-case keys; it is, however, a deprecated approach.

Table 22.7. "data" elements

key	description
JMS_MSG	instance of javax.jms.Message
JMS_DATA_STREAM	Instance of java.io.InputStream. Accessible only for TextMessage, BytesMessage, StreamMessage, ObjectMessage (only if payload object is instance of String). Strings are encoded in UTF-8.
JMS_DATA_TEXT	Instance of String. Only for TextMessage and ObjectMessage, where payload object is instance of String.
JMS_DATA_OBJECT	Instance of java.lang.Object - message payload. Only for ObjectMessage.

The “Data” container is passed to a task that can use it, depending on its implementation. For example, the task "execute graph" passes it to the executed graph as “dictionary entries.”

In the Cluster environment, you can specify explicitly node IDs, which can execute the task. However, if the “data” payload is not serializable and the receiving and executing node differ, an error will be thrown as the Cluster cannot pass the “data” to the executing node.

Inside a graph or a jobflow, data passed as dictionary entries can be used in some component attributes. For example, a File URL attribute would look like: "dict:JMS_DATA_STREAM:discrete" for reading the data directly from the incoming JMS message using a proxy stream.

For backwards compatibility, all listed dictionary entries can also be accessed using lower-case keys; it is, however, a deprecated approach.

Universal event listeners

Since 2.10

Universal Event Listeners allow you to write a piece of Groovy code that controls when an event is triggered, subsequently executing a predefined task. The Groovy code is periodically executed and when it returns TRUE, the task is executed.

Table 22.8. Attributes of Universal message task

Attribute	Description
Node IDs to handle the event	In the Cluster environment, each universal event listener has a "Node IDs" attribute which may be used for specification which cluster node will perform the Groovy code. There are following possibilities: • No failover: Just one node ID specified - Only specified node performs the Groovy code, however node status must be "ready". When the node isn't ready, code isn't performed at all. • Failover with node concurrency: No node ID specified (empty input) - All cluster nodes with status "ready" concurently perform Groovy code. So the code is executed on each node in specified interval. • Failover with node reservation: More node IDs specified (separated by comma) - Just one of specified nodes performs groovy code. If the node fails from any reason (or its status isn't "ready"), any other "ready" node from the list continues with periodical groovy code processing. In standalone environment, the "Node IDs" attribute is ignored.
Interval of check in seconds	Periodicity of Groovy code execution.
Groovy code	Groovy code that evaluates either to TRUE (execute the task) or FALSE (no action). See below for more details.

Groovy code

A piece of Groovy is repeatedly executed and evaluated; based on the result, the event is either triggered and the task executed or no action is taken.

For example, you can continually check for essential data sources before starting a graph. Or, you can do complex checks of a running graph and, for example, decide to kill it if necessary. You can even call the CloverETL Server core functions using the ServerFacade interface, see Javadoc: http://host:port/clover/javadoc/index.html

Evaluation Critieria

If the Groovy code returns Boolean.TRUE, the event is triggered and the associated task is executed. Otherwise, nothing happens.

If the Groovy code throws an exception, it's considered a coding error and the event is NOT triggered. Thus, exceptions should be properly handled in the Groovy code.

Table 22.9. Variables accessible in groovy code

type	key	description
java.util.Properties	properties	Empty container which may be filled with String-String key-value pairs in your Groovy code. It is passed to the task which may use them somehow. I.e. task "execute graph" passes these parameters to the executed graph.
java.util.Map<String, Object>	data	Empty container which may be filled with String-Object key-value pairs in your Groovy code. It is passed to the task which may use them somehow according to its implementation. I.e. task "execute graph" passes it to the executed graph as "dictionary entries". Please note that it is not serializable, thus if the task is relying on it, it can be processed properly only on the same cluster node.
javax.servlet.ServletContext	servletContext	instance of ServletContext
com.cloveretl.server.api.ServerFacade	serverFacade	instance of serverFacade usable for calling CloverETL Server core features.
String	sessionToken	sessionToken, needed for calling serverFacade methods

File event listeners

Since 1.3

File Event Listeners allow you to monitor changes on a specific file system path – for example, new files appearing in a folder – and react to such an event with a predefined task.

You can either specify an exact path or use a wildcard, then set a checking interval in seconds, and finally, define a task to process the event.

There is a global minimal check interval that you can change if necessary in the configuration ("clover.event.fileCheckMinInterval" property).

In the Cluster environment, each file event listener has a "Node IDs" attribute which may be used for specification which cluster node will perform the checks on its local file system. There are following possibilities:

- No failover: Just one node ID specified - Only specified node checks its local filesystem, however node status must be "ready". When the node isn't ready, file system isn't checked at all.
- Failover with node concurrency: No node ID specified (empty input) - All cluster nodes with status "ready" concurently check their local filesystem according to file event listener attributes settings
- Failover with node reservation: More node IDs specified (separated by comma) - Just one of specified nodes checks its filesystem. If the node fails from any reason (or its status isnt "ready"), any other "ready" node from

the list continues with checking on its filesystem. Please note, that when file event listener is re-initialized on another cluster node, it compares last directory content on the failed node's filesystem with the its own local filesystem.

In standalone environment, the "Node IDs" attribute is ignored.

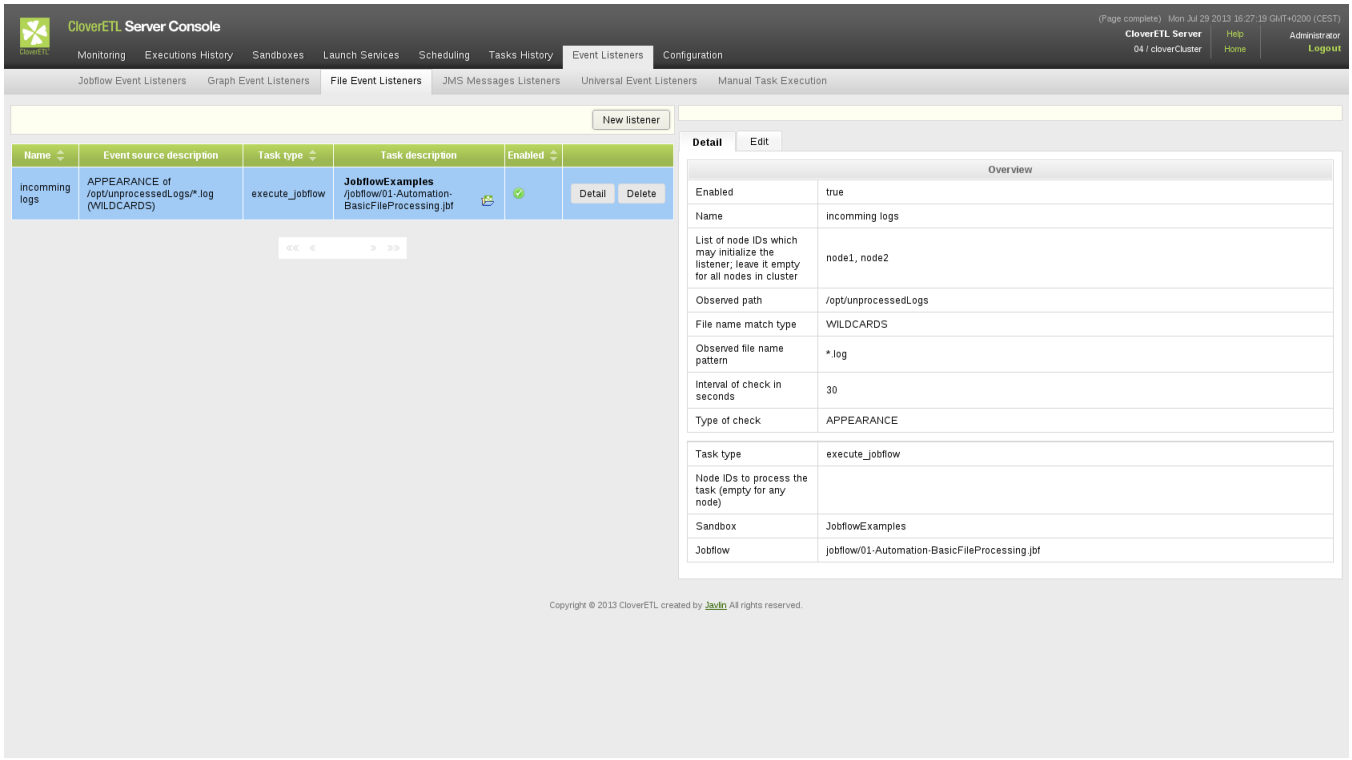


Figure 22.9. Web GUI - "File event listeners" section

Observed file

Observed file is specified by directory path and file name pattern.

User may specify just one exact file name or file name pattern for observing more matching files in specified directory. If there are more changed files matching the pattern, separated event is triggered for each of these files.

There are three ways how to specify file name pattern of observed file(s)

- Exact match (p. 153)
- Wildcards (p. 153)
- Regular expression (p. 154)

Exact match

You specify exact name of the observed file.

Wildcards

You can use wildcards common in most operating systems (*, ?, etc.)

- * - Matches zero or more instances of any character
- ? - Matches one instance of any character

- [...] - Matches any of characters enclosed by the brackets
- \ - Escape character

Examples

- *.csv - Matches all CSV files
- input_*.csv - Matches i.e. input_001.csv, input_9.csv
- input_???.csv - Matches i.e. input_001.csv, but does not match input_9.csv

Regular expression

Examples

- (.*?)\.(jpg|jpeg|png|gif)$ - Matches image files

Notes

- It is strongly recommended to use absolute paths. It is possible to use relative path, but working directory depends on application server.
- Use forward slashes as file separators, even on MS Windows OS. Backslashes might be evaluated as escape sequences.

File Events

For each listener you have to specify event type, which you are interested in.

There are four types of file events:

- file APPEARANCE (p. 154)
- file DISAPPEARANCE (p. 154)
- file SIZE (p. 154)
- file CHANGE_TIME (p. 155)

file APPEARANCE

Event of this type occurs, when the observed file is created or copied from another location between two checks. Please keep in mind, that event of this type occurs immediately when new file is detected, regardless it is complete or not. Thus task which may need complete file is executed when file is still incomplete. Recommended approach is to save file to the different location and when it is complete, move/rename to observed location where CloverETL Server may detect it. File moving/renaming should be atomic operation.

Event of this type does not occur when the file has been updated (change of timestamp or size) between two checks. Appearance means that the file didn't exist during previous check and it exists now, during current check.

file DISAPPEARANCE

Event of this type occurs, when observed file is deleted or moved to another location between two checks.

file SIZE

Event of this type occurs when the size of the observed file has changed between two checks. Event of this type is never produced when file is created or removed. File must exist during both checks.

file CHANGE_TIME

Event of this type occurs, when change time of observed file has changed between two checks. Event of this type is never produced when file is created or removed. File must exist during both checks.

Check interval, Task and Use cases

- User may specify minimal time interval between two checks. It is specified in seconds.
- Each listener defines task, which will be processed as the reaction for file event. All task types and theirs attributes are described in section Scheduling and GraphEventListeners
- • Graph Execution, when file with input data is accessible
  - Graph Execution, when file with input data is updated
  - Graph Execution, when file with generated data is removed and must be recreated

How to use source of event during task processing

File, which caused event (considered as source of event) may be used during task processing. i.e. reader/writer components in graph transformations may refer to this file by special placeholders: `${EVENT_FILE_PATH}` - path to directory which contains event source `${EVENT_FILE_NAME}` - name of event source.

Please note that previous versions used lower-case placeholders. Since version 3.3, placeholders are upper-case, however lower-case still work for backward compatibility.

i.e. if event source is: `/home/clover/data/customers.csv`, placeholders will contain: EVENT_FILE_PATH - `/home/clover/data`, EVENT_FILE_NAME - `customers.csv`

For "graph execution" task this works only if the graph is not pooled. Thus "keep in pool interval" must be set to 0 (default value).

Chapter 23. API

Simple HTTP API

The Simple HTTP API is a basic Server automation tool that lets you control the Server from external applications using simple HTTP calls.

Most of operations is accessible using the HTTP GET method and return plain text. Thus, both “request” and “response” can be conveniently sent and parsed using very simple tools (wget, grep, etc.).

If global security is “on” (on by default), Basic HTTP authentication is used. Authenticated operations will require valid user credentials with corresponding permissions.

Note that the ETL graph-related operations "graph_run", "graph_status" and "graph_kill" also work for jobflows and Data Profiler jobs.

The generic pattern for a request URL:

```
http://[domain]:[port]/[context]/[servlet]/[operation]?[param1]=[value1]&[param2]=[value2]...
```

For a wget client, you can use following command line:

```
wget --user=$USER --password=$PASS -O ./$OUTPUT_FILE $REQUEST_URL
```


Operation help

parameters

no

returns

list of possible operations and parameters with its descriptions

example

```
http://localhost:8080/clover/request_processor/help
```

Operation graph_run

Call this operation to start execution of the specified job. Operation is called graph_run for backward compatibility, however it may execute ETL graph, jobflow or profiler job.

parameters

Table 23.1. Parameters of graph_run

parameter name	mandatory	default	description
graphID	yes	-	File path to the job file, relative to the sandbox root.
sandbox	yes	-	Text ID of sandbox.
additional job parameters	no		Any URL parameter with "param_" prefix is passed to executed job and may be used in transformation XML as placeholder, but without "param_" prefix. e.g. "param_FILE_NAME" specified in URL may be used in the XML as ${FILE_NAME}. These parameters are resolved only during loading of XML, so it cannot be pooled.
additional config parameters	no		URL Parameters prefixed with "config_" can set some of the execution parameters. For graphs, the following parameters are supported: • `config_skipCheckConfig` - when set to "false", graph configuration will be checked before the execution. • `config_logLevel` - log level of the executed graph, one of OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL. • `config_clearObsoleteTempFiles` - when set to "true", temp files of previous runs of this graph will be deleted before the execution. • `config_debugMode` - when set to "true", debug mode for given graph will be enabled. See Job config properties (p. 101) for more info.
nodeID	no	-	In cluster mode it's ID of node which should execute the job. However it's not final. If the graph is distributed, or the node is disconnected, the graph may be executed on some another node.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

run ID: incremental number, which identifies each execution request

example

```
http://localhost:8080/clover/request_processor/graph_run?graphID=graph/graphDBExecute.grf&sandbox=mva
```

Operation graph_status

Call this operation to obtain status of specified job execution. Operation is called graph_status for backward compatibility, however it may return status of ETL graph or jobflow.

parameters

Table 23.2. Parameters of graph_status

parameter name	mandatory	default	description
runID	yes	-	Id of each graph execution
returnType	no	STATUS	STATUS | STATUS_TEXT | DESCRIPTION | DESCRIPTION_XML
waitForStatus	no	-	Status code which we want to wait for. If it is specified, this operation will wait until graph is in required status.
waitTimeout	no	0	If waitForStatus is specified, it will wait only specified amount of milliseconds. Default 0 means forever, but it depends on application server configuration. When the specified timeout expires and graph run still isn't in required status, server returns code 408 (Request Timeout). 408 code may be also returned by application server if its HTTP request timeout expires before.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Status of specified graph. It may be number code, text code or complex description in dependence of optional parameter returnType. Description is returned as plain text with pipe as separator, or as XML. Schema describing XML format of the XML response is accessible on CloverETL Server URL: http://[host]:[port]/clover/schemas/executions.xsd In dependence on waitForStatus parameter may return result immediately or wait for specified status.

example

```
http://localhost:8080/clover/request_processor/graph_status ->
           -> ?runID=123456&returnType=DESCRIPTION&waitForStatus=FINISHED&waitTimeout=60000
```

Operation graph_kill

Call this operation to abort/kill job execution. Operation is called graph_kill for backward compatibility, however it may abort/kill ETL graph, jobflow or profiler job.

parameters

Table 23.3. Parameters of graph_kill

parameter name	mandatory	default	description
runID	yes	-	Id of each graph execution
returnType	no	STATUS	STATUS | STATUS_TEXT | DESCRIPTION
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Status of specified graph after attempt to kill it. It may be number code, text code or complex description in dependence of optional parameter.

example

```
http://localhost:8080/clover/request_processor/graph_kill?runID=123456&returnType=DESCRIPTION
```

Operation server_jobs

parameters

no

returns

List of runIDs currently running jobs.

example

```
http://localhost:8080/clover/request_processor/server_jobs
```

Operation sandbox_list

parameters

no

returns

List of all sandbox text IDs. In next versions will return only accessible ones.

example

```
http://localhost:8080/clover/request_processor/sandbox_list
```

Operation sandbox_content

parameters

Table 23.4. Parameters of sandbox_content

parameter name	mandatory	default	description
sandbox	yes	-	text ID of sandbox
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

List of all elements in specified sandbox. Each element may be specified as file path relative to sandbox root.

example

```
http://localhost:8080/clover/request_processor/sandbox_content?sandbox=mva
```

Operation executions_history

parameters

Table 23.5. Parameters of executions_history

parameter name	mandatory	default	description
sandbox	yes	-	text ID of sandbox
from	no		Lower datetime limit of start of execution. Operation will return only records after (and equal) this datetime. Format: "yyyy-MM-dd HH:mm" (must be URL encoded).
to	no		Upper datetime limit of start of execution. Operation will return only records before (and equal) this datetime. Format: "yyyy-MM-dd HH:mm" (must be URL encoded).
stopFrom	no		Lower datetime limit of stop of execution. Operation will return only records after (and equal) this datetime. Format: "yyyy-MM-dd HH:mm" (must be URL encoded).
stopTo	no		Upper datetime limit of stop of execution. Operation will return only records before (and equal) this datetime. Format: "yyyy-MM-dd HH:mm" (must be URL encoded).
status	no		Current execution status. Operation will return only records with specified STATUS. Meaningful values are RUNNING | ABORTED | FINISHED_OK | ERROR
sandbox	no		Sandbox code. Operation will return only records for graphs from specified sandbox.
graphId	no		Text Id, which is unique in specified sandbox. File path relative to sandbox root
orderBy	no		Attribute for list ordering. Possible values: id | graphId | status | startTime | stopTime. There is no ordering by default.
orderDescend	no	true	Switch which specifies ascending or descending ordering. If it is true (which is default), ordering is descending.
returnType	no	IDs	Possible values are: IDs | DESCRIPTION | DESCRIPTION_XML
index	no	0	Index of the first returned records in whole record set. (starting from
records	no	infinite	Max amount of returned records.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

List of executions according to filter criteria.

For `returnType==IDs` returns simple list of runIDs (with new line delimiter).

For `returnType==DESCRIPTION` returns complex response which describes current status of selected executions, their phases, nodes and ports.

```
execution|[runID]|[status]|[username]|[sandbox]|[graphID]|[startedDatetime]|[finishedDatetime]|[clusterNode]|[grap
phase|[index]|[execTimeInMilis]
node|[nodeID]|[status]|[totalCpuTime]|[totalUserTime]|[cpuUsage]|[peakCpuUsage]|[userUsage]|[peakUserUsage]
port|[portType]|[index]|[avgBytes]|[avgRows]|[peakBytes]|[peakRows]|[totalBytes]|[totalRows]
```

example of request

```
http://localhost:8080/clover/request_processor/executions_history ->
```

```
              -> ?from=&to=2008-09-16+16%3A40&status=&sandbox=def&graphID=&index=&records=&returnType=DESCRIPTION
```

example of DESCRIPTION (plain text) response

```
execution|13108|FINISHED_OK|clover|def|test.grf|2008-09-16 11:11:19|2008-09-16 11:11:58|nodeA|2.4
phase|0|38733
node|DATA_GENERATOR1|FINISHED_OK|0|0|0.0|0.0|0.0|0.0
port|Output|0|0|0|0|130|10
node|TRASH0|FINISHED_OK|0|0|0.0|0.0|0.0|0.0
port|Input|0|0|0|5|0|130|10
node|SPEED_LIMITER0|FINISHED_OK|0|0|0.0|0.0|0.0|0.0
port|Input|0|0|0|0|0|130|10
port|Output|0|0|0|5|0|130|10
execution|13107|ABORTED|clover|def|test.grf|2008-09-16 11:11:19|2008-09-16 11:11:30
phase|0|11133
node|DATA_GENERATOR1|FINISHED_OK|0|0|0.0|0.0|0.0|0.0
port|Output|0|0|0|0|0|130|10
node|TRASH0|RUNNING|0|0|0.0|0.0|0.0|0.0
port|Input|0|5|0|5|0|52|4
node|SPEED_LIMITER0|RUNNING|0|0|0.0|0.0|0.0|0.0
port|Input|0|0|0|0|0|130|10
port|Output|0|5|0|5|0|52|4
```

For `returnType==DESCRIPTION_XML` returns complex data structure describing one or more selected executions in XML format. Schema describing XML format of the XML response is accessible on CloverETL Server URL: http://[host]:[port]/clover/schemas/executions.xsd

Operation suspend

Suspends server or sandbox (if specified). Suspension means, that no graphs may me executed on suspended server/sandbox.

parameters

Table 23.6. Parameters of suspend

parameter name	mandatory	default	description
sandbox	no	-	Text ID of sandbox to suspend. If not specified, it suspends whole server.
atonce	no		If this param is set to true, running graphs from suspended server (or just from sandbox) are aborted. Otherwise it can run until it is finished in common way.

returns

Result message

Operation resume

parameters

Table 23.7. Parameters of resume

parameter name	mandatory	default	description
sandbox	no	-	Text Id of sandbox to resume. If not specified, server will be resumed.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Result message

Operation sandbox_create

This operation creates specified sandbox. If it's sandbox of "partitioned" or "local" type, create also locations by "sandbox_add_location" operation.

parameters

Table 23.8. Parameters of sandbox create

parameter name	mandatory	default	description
sandbox	yes	-	Text Id of sandbox to be created.
path	no	-	Path to the sandbox root if server is running in standalone mode.
type	no	shared	Sandbox type: shared | partitioned | local. For standalone server may be left empty, since the default "shared" is used.
createDirs	no	true	Switch whether to create directory structure of the sandbox (only for standalone server or "shared" sandboxes in cluster environment).
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Result message

Operation sandbox_add_location

This operation adds location to specified sandbox. Only useable for sandboxes of type partitioned or local.

parameters

Table 23.9. Parameters of sandbox add location

parameter name	mandatory	default	description
sandbox	yes	-	Sandbox which we want to add location to.
nodeId	yes	-	Location attribute - node which has direct access to the location.
path	yes	-	Location attribute - path to the location root on the specified node.
location	no	-	Location attribute - location storage ID. If it's not specified, new one will be generated.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Result message

Operation sandbox_remove_location

This operation removes location from specified sandbox. Only sandboxes of type partitioned or local can have locations asociated.

parameters

Table 23.10. Parameters of sandbox add location

parameter name	mandatory	default	description
sandbox	yes	-	Removes specified location from its sandbox.
location	yes	-	Location storage ID. If the specified location isn't attached to the specified sandbox, sandbox won't be changed.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Result message

Operation download_sandbox_zip

This operation downloads content of specified sandbox as an ZIP archive.

parameters

Table 23.11. Parameters

parameter name	mandatory	default	description
sandbox	yes	-	Code of the sandbox to be downloaded.

returns

Content of the specified sandbox as a ZIP archive

example

```
wget --http-user=username --http-password=password http://localhost:8080/clover/simpleHttpApi/download_sandbox_zi
```

Operation upload_sandbox_zip

This operation uploads content of a ZIP archive into specified sandbox.

parameters

Table 23.12. Parameters

parameter name	mandatory	default	description
sandbox	yes	-	Code of the sandbox the ZIP file will be expanded to.
zipFile	yes	-	The ZIP archive file.
overwriteExisting	no	false	If `true`, the files already present in the sandbox will be overwritten.
deleteMissing	no	false	If `true`, the files not present in the ZIP file will be deleted from sandbox.
fileNameEncoding	no	UTF-8	The encoding that was used to store file names in the ZIP archive.

returns

Result message

example of request (with using curl CLI tool (http://curl.haxx.se/))

```
curl -u username:password -F "overwriteExisting=true"
    -F "zipFile=@/tmp/my-sandbox.zip"
    http://localhost:8080/clover/simpleHttpApi/upload_sandbox_zip
```

Operation cluster_status

This operation displays cluster's nodes list.

parameters

no

returns

Cluster's nodes list.

Operation export_server_config

This operation exports current server configuration in XML format.

parameters

Table 23.13. Parameters of server configuration export

parameter name	mandatory	default	description
include	no	all	Selection which items will be included in the exported XML file; the parameter may be specified multiple times. Possible values are: • `all` - include items of all types • `users` - include list of users • `userGroups` - include list of user groups • `sandboxes` - include list of sandboxes • `jobConfigs` - include list of job configuration parameters • `schedules` - include list of schedules • `eventListeners` - include list of event listeners • `launchServices` - include list of launch services • `tempSpaces` - include list of temporary spaces

returns

Current server configuration as an XML file.

example

```
wget http://localhost:8080/clover/simpleHttpApi/export_server_config
```

Operation import_server_config

This operation imports server configuration.

parameters

Table 23.14. Parameters of server configuration import

parameter name	mandatory	default	description
xmlFile	yes	-	An XML file with server's configuration.
dryRun	no	true	If `true`, a dry run is performed with no actual changes written.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should the response be: MESSAGE for simple message, FULL for full XML report.
newOnly	no	false	If `true` only new items will imported to the server; the items already present on the server will be left untouched.
include	no	all	Selection which items will be imported from the XML; the parameter may be specified multiple times. Possible values are: • `all` - import items of all types • `users` - import users • `userGroups` - import user groups • `sandboxes` - import sandboxes • `jobConfigs` - import job configuration parameters • `schedules` - import schedules • `eventListeners` - import listeners • `launchServices` - import launch services • `tempSpaces` - import temporary spaces

returns

Result message or XML report

example of request (with using curl CLI tool (http://curl.haxx.se/))

```
curl -u username:password -F "dryRun=true" -F "verbose=FULL"
    -F "xmlFile=@/tmp/clover_configuration_2013-07-10_14-03-23+0200.xml"
    http://localhost:8080/clover/simpleHttpApi/import_server_config
```

JMX mBean

The CloverETL Server JMX mBean is an API that you can use for monitoring the internal status of the Server.

MBean is registered with the name:

```
com.cloveretl.server.api.jmx:name=cloverServerJmxMBean
```

.

JMX configuration

Application's JMX MBeans aren't accessible outside of JVM by default. It needs some changes in application server configuration to make them accessible.

This section describes how to configure JMX Connector for development and testing. Thus authentication may be disabled. For production deployment authentication should be enabled. Please refer further documentation to see how to achieve this. i.e. http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html#auth

Configurations and possible problems:

- How to configure JMX on Apache Tomcat (p. 166)
- How to configure JMX on Glassfish (p. 167)
- How to configure JMX on WebSphere (p. 167)
- Possible problems (p. 168)

How to configure JMX on Apache Tomcat

Tomcat's JVM must be executed with these self-explanatory parameters:

1. `-Dcom.sun.management.jmxremote=true`
2. `-Dcom.sun.management.jmxremote.port=8686`
3. `-Dcom.sun.management.jmxremote.ssl=false`
4. `-Dcom.sun.management.jmxremote.authenticate=false`
5. `-Djava.rmi.server.hostname=your.server.domain (necessary only for remote JMX connections)`

On UNIX like OS set environment variable CATALINA_OPTS i.e. like this:

```
export CATALINA_OPTS="-Dcom.sun.management.jmxremote=true
                      -Dcom.sun.management.jmxremote.port=8686
                      -Dcom.sun.management.jmxremote.ssl=false
                      -Dcom.sun.management.jmxremote.authenticate=false
                      -Djava.rmi.server.hostname=your.server.domain.com"
```

File TOMCAT_HOME/bin/setenv.sh (if it does not exist, you may create it) or TOMCAT_HOME/bin/catalina.sh

On Windows it might be tricky, that each parameter must be set separately:

```
set CATALINA_OPTS=-Dcom.sun.management.jmxremote=true
set CATALINA_OPTS=%CATALINA_OPTS% -Dcom.sun.management.jmxremote.port=8686
set CATALINA_OPTS=%CATALINA_OPTS% -Dcom.sun.management.jmxremote.authenticate=false
set CATALINA_OPTS=%CATALINA_OPTS% -Dcom.sun.management.jmxremote.ssl=false
```

```
set CATALINA_OPTS=%CATALINA_OPTS% -Djava.rmi.server.hostname=your.server.domain
```

File TOMCAT_HOME/bin/setenv.bat (if it does not exist, you may create it) or TOMCAT_HOME/bin/catalina.bat

With these values, you can use URL

```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

for connection to JMX server of JVM. No user/password is needed

How to configure JMX on Glassfish

Go to Glassfish admin console (by default accessible on http://localhost:4848 with admin/adminadmin as user/password)

Go to section "Configuration" > "Admin Service" > "system" and set attributes like this:

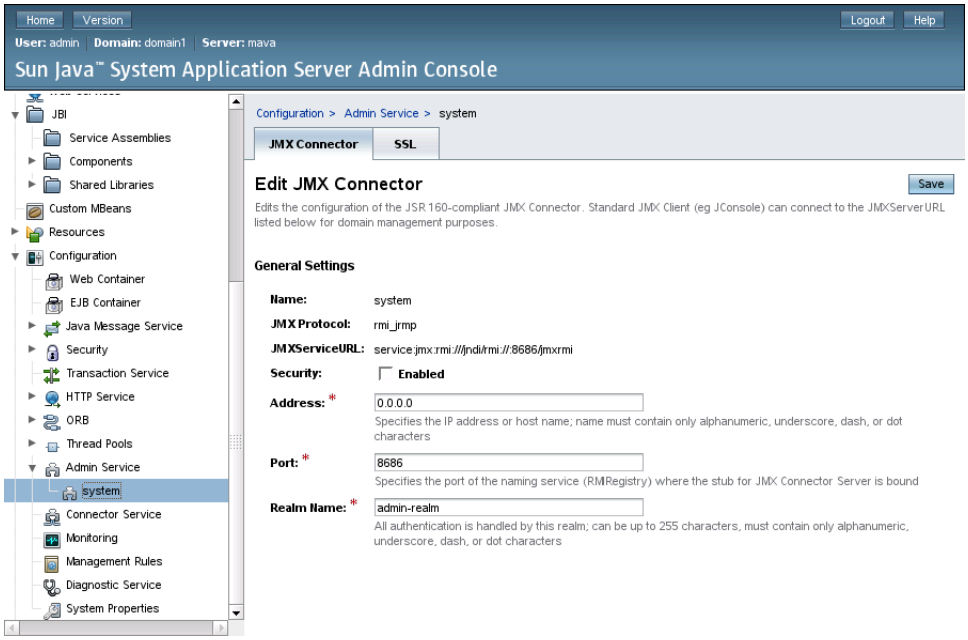


Figure 23.1. Glassfish JMX connector

With these values, you can use URL

```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

for connection to JMX server of JVM.

Use admin/adminadmin as user/password. (admin/adminadmin are default glassfish values)

How to configure JMX on WebSphere

WebSphere does not require any special configuration, but the clover MBean is registered with the name, that depends on application server configuration:

```
com.cloveretl.server.api.jmx:cell=[cellName],name=cloverServerJmxMBean,node=[nodeName],
            process=[instanceName]
```

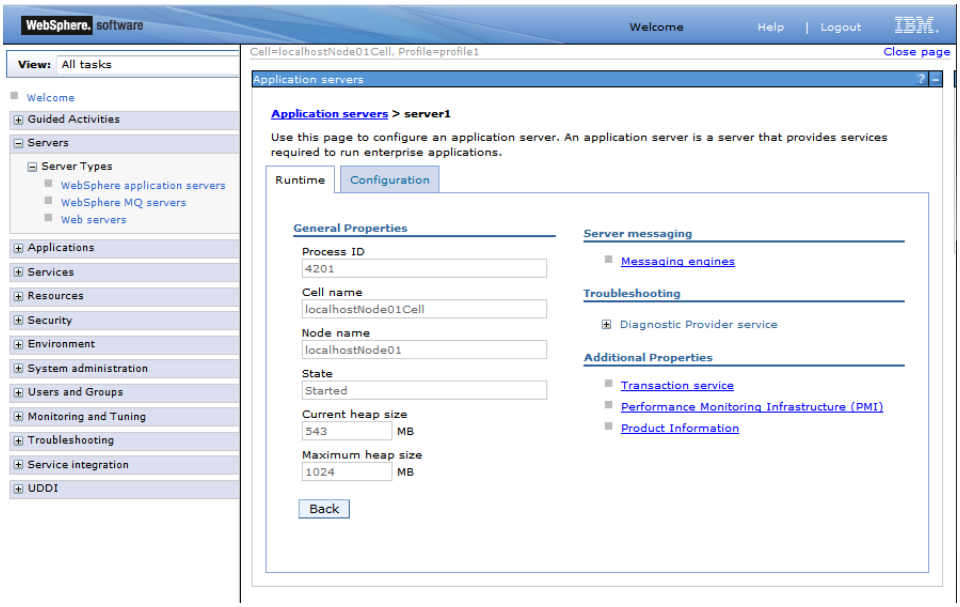


Figure 23.2. WebSphere configuration

URL for connecting to JMX server is:

```
service:jmx:iiop://[host]:[port]/jndi/JMXConnector
```

where *host* is the host name you are connectiong to and *port* is RMI port number. If you have a default WebSphere installation, the JNDI port number will likely be 9100, depending on how many servers there are installed on one system and the specific one you want to connect to. To be sure, when starting WebSphere, check the logs, as it will dump a line like

```
0000000a RMIConnectorC A   ADMC0026I: The RMI Connector is available at port 9100
```

You will also need to set on the classpath following jar files from WebSphere home directory:

runtimes/com.ibm.ws.admin.client_8.5.0.jar
runtimes/com.ibm.ws.ejb.thinclient_8.5.0.jar
runtimes/com.ibm.ws.orb_8.5.0.jar

Possible problems

- Default JMX mBean server uses RMI as a transport protocol. Sometimes RMI cannot connect remotely when one of peers uses Java version 1.6. Solution is quite easy, just set these two system properties: `-Djava.rmi.server.hostname=[hostname or IP address] -Djava.net.preferIPv4Stack=true`

Operations

For details about operations please see the JavaDoc of the MBean interface:

JMX API MBean JavaDoc is accessible in the running CloverETL Server instance on URL: http://[host]:[port]/[contextPath]/javadoc-jmx/index.html

SOAP WebService API

The CloverETL Server SOAP Web Service is an advanced API that provides an automation alternative to the Simple HTTP API. While most of the HTTP API operations are available in the SOAP interface too (though not all of them), the SOAP API provides additional operations for manipulating sandboxes, monitoring, etc.

The SOAP API service is accessible on the following URL:

```
http://[host]:[port]/clover/webservice
```

The SOAP API service descriptor is accessible on URL:

```
http://[host]:[port]/clover/webservice?wsdl
```

Protocol HTTP can be changed to secured HTTPS based on the web server configuration.

SOAP WS Client

Exposed service is implemented with the most common binding style "document/literal", which is widely supported by libraries in various programming languages.

To create client for this API, only WSDL document (see the URL above) is needed together with some development tools according to your programming language and development environments.

JavaDoc of WebService interface with all related classes is accessible in the running CloverETL Server instance on URL http://[host]:[port]/[contextPath]/javadoc-ws/index.html

If the web server has HTTPS connector configured, also the client must meet the security requirements according to web server configuration. i.e. client trust + key stores configured properly

SOAP WS API authentication/authorization

Since exposed service is stateless, authentication "sessionToken" has to be passed as parameter to each operation. Client can obtain authentication sessionToken by calling "login" operation.

Launch Services

Launch Services allow you to publish a transformation graph or a jobflow as a Web Service. With Launch Services, CloverETL transformations can be exposed to provide request-response based data interface (e.g. searches, complicated lookups, etc.) for other application or directly to users.

Launch Services Overview

The architecture of a Launch Service is relatively simple, following the basic design of multi-tiered applications utilizing a web browser.

For example, you can build a user-friendly form that the user fills in and sends to the CloverETL Server for processing.

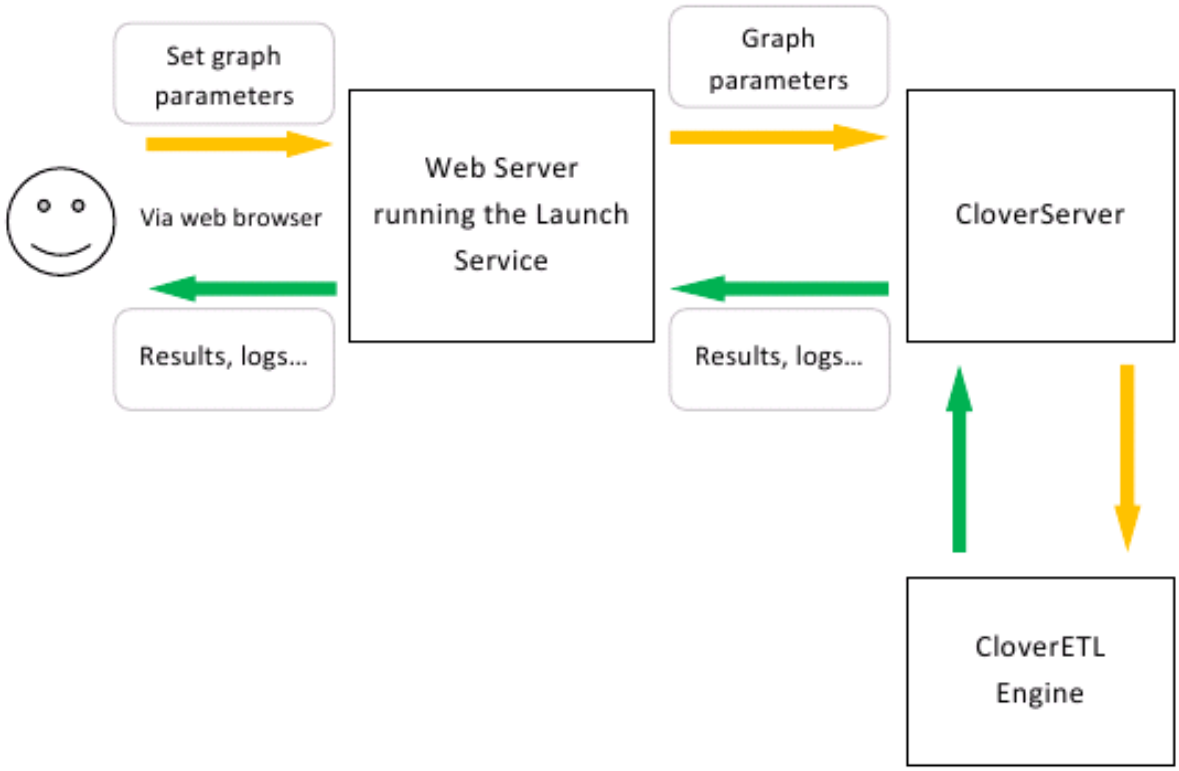


Figure 23.3. Launch Services and CloverETL Server as web application back-end

Deploying Graph in Launch Service

To prepare a graph for publishing as a Launch Service, keep this in mind during the design process:

1. You can define a graph/jobflow listeners to create parameterized calls. Parameters are passed to the graph as Dictionary entries – so, design the graph so that it uses the Dictionary as input/output for parameters (e.g. file names, search terms, etc.)

2. The graph will need to be published in the Launch Services section, where you provide the configuration and binding for parameters to dictionary entries.

Using Dictionary in ETL Graph/Jobflow for a Launch Services

A graph or a jobflow published as a service usually means that the caller sends request data (parameters or data) and the transformation processes it and returns back the results.

In a Launch Service definition, you can bind a service's parameters to Dictionary entries. These need to be predefined in the transformation.

Dictionary is a key-value temporary data interface between the running transformation and the caller. Usually, although not restricted to, Dictionary is used to pass parameters in and out the executed transformation.

For more information about Dictionary, read the "Dictionary" section in the CloverETL Designer User's Guide.

Configuring the job in CloverETL Server Web GUI

Each Launch Service configuration is identified by its name, user, and group restriction. You can create several configurations with the same name, which is valid as long as they differ in their user or group restrictions.

User restrictions can then be used to launch different jobs for different users, even though they use the same launch configuration (i.e. name). For example, developers may want to use a debug version of the job, while the end customers will want to use a production job. The user restriction can also be used to prohibit certain users from executing the launch configuration completely.

Similarly, a group restriction can be used to differentiate jobs based on the user's group membership.

If multiple configurations match the current user/group and configuration name, the most specific one is picked. (The user name has higher priority than the group name.)

Adding New Launch Configuration

Use the "New launch configuration" button on the Launch Services tab to create a new Launch Service.

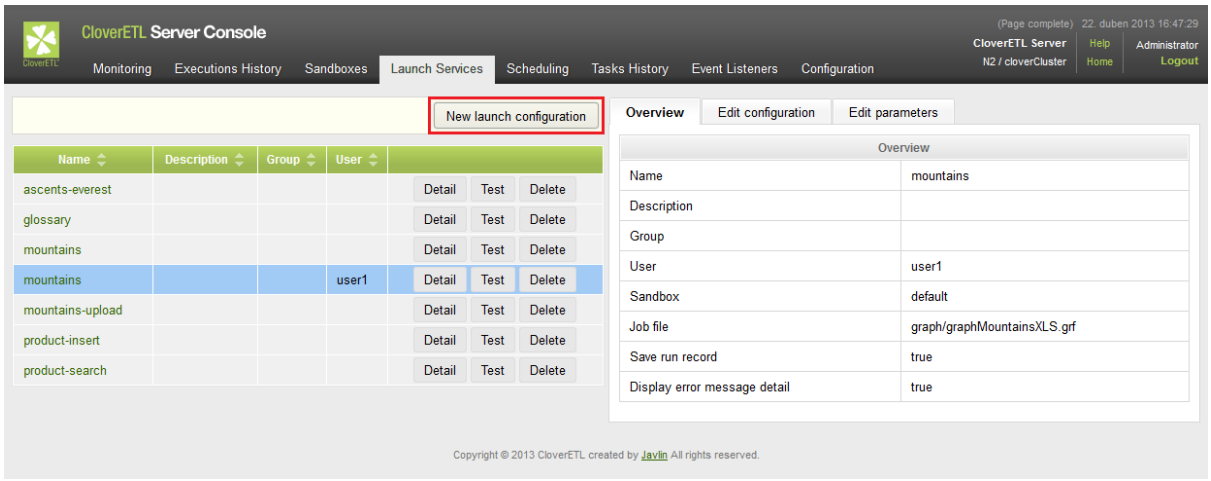


Figure 23.4. Launch Services section

The name is the identifier for the service and will be used in the service URL. Then, select a sandbox and either a transformation graph or a jobflow that you want to publish.

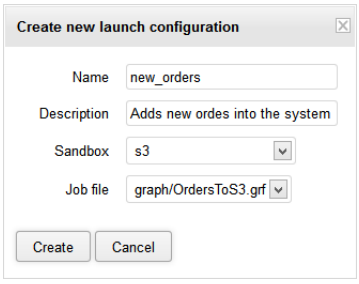


Figure 23.5. Creating a new launch configuration

Once you create the new Launch Service, you can set additional attributes like:

1. User and group access restrictions and additional configuration options (Edit Configuration)

2. Bind Launch Service parameters to Dictionary entries (Edit Parameters)

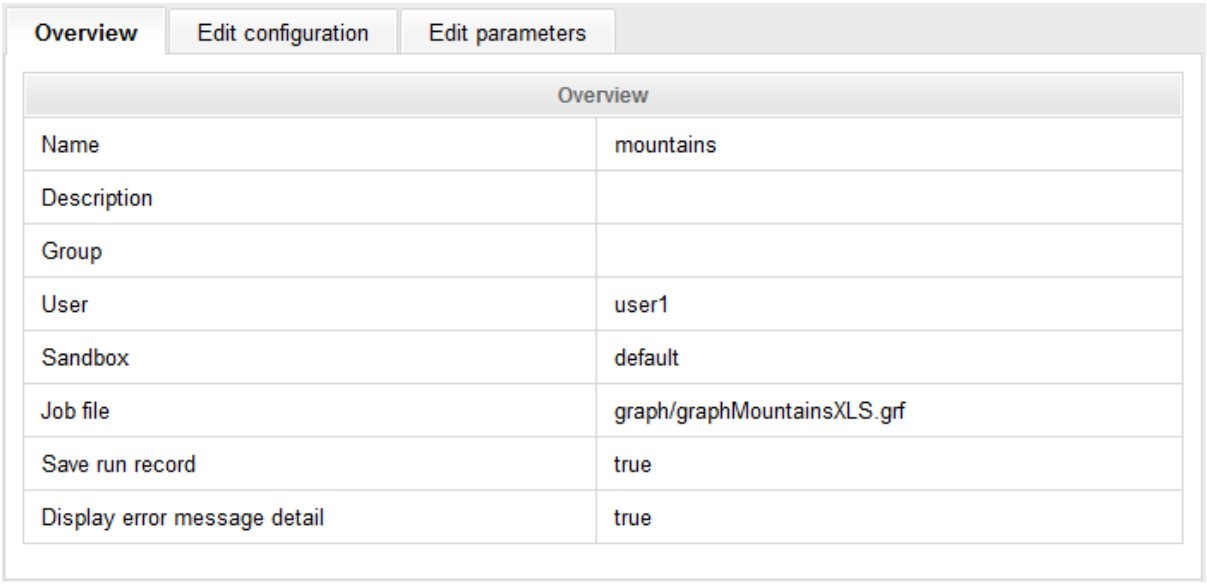


Figure 23.6. Overview tab

The Overview tab shows the basic details about the launch configuration. These can be modified in the Edit Configuration tab:

Edit configuration

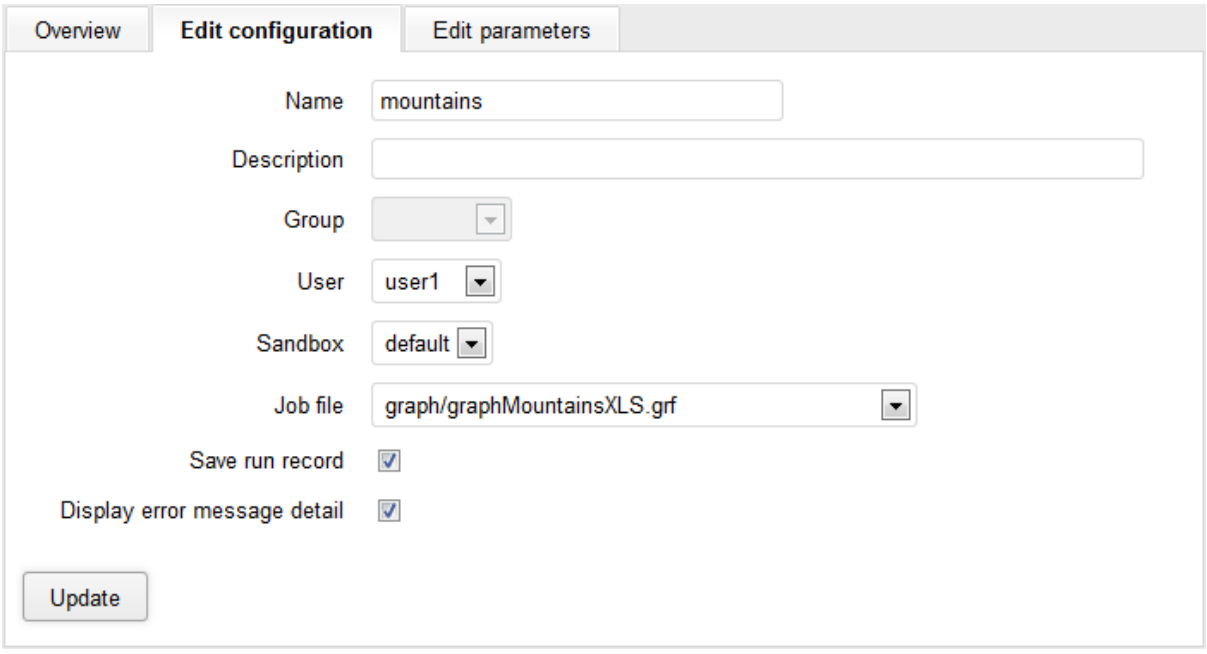


Figure 23.7. Edit Configuration tab

Editing configurations:

- *Name* - The name (identifier) under which the configuration will be accessible from the web.
- *Description* - The description of the configuration.
- *Group* - Restricts the configuration to a specific group of users.
- *User* - Restricts the configuration to a specified user.

- *Sandbox* - The CloverETL Sandbox where the configuration will be launched.
- *Job file* - Selects the job to run.
- *Save run record* - If checked, the details about the launch configuration will be stored in the Execution History. Uncheck this if you need to increase performance – storing a run record decreases response times for high frequency calls.
- *Display error message detail* - Check this if you want to get a verbose message in case the launch fails.

Edit parameters

The “Edit parameters” tab can be used to configure parameter mappings for the launch configuration. The mappings are required for the Launch Service to be able to correctly assign parameters values based on the values sent in the launch request.

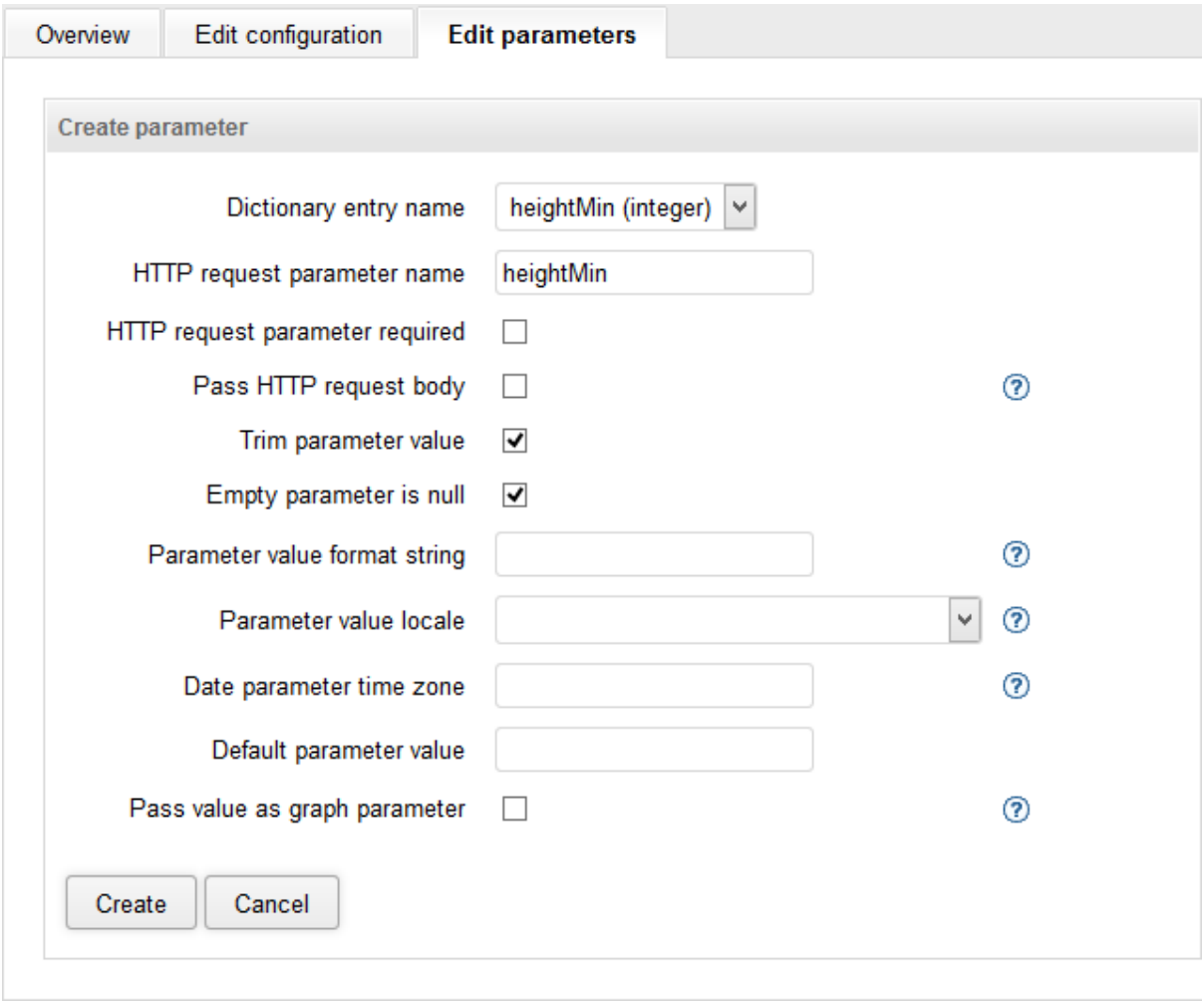


Figure 23.8. Creating new parameter

To add a new parameter binding, click on the “Add parameter” button. Every required a graph/jobflow listenerproperty defined by the job needs to be created here.

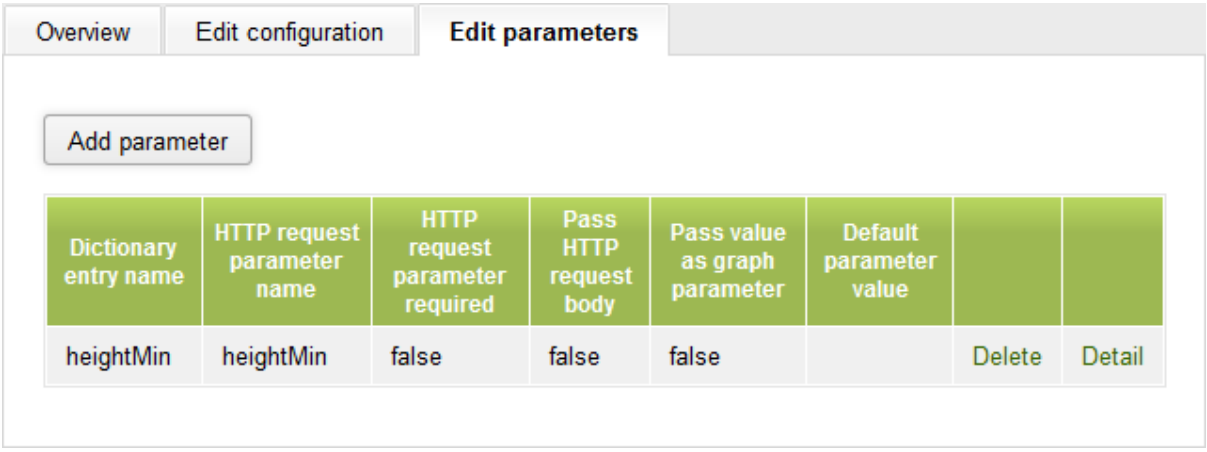


Figure 23.9. Edit Parameters tab

You can set the following fields for each property:

- *Dictionary entry name* - The name of the Dictionary entry defined in the graph/jobflow that you want to bind.
- *HTTP request parameter name* - The name of this property as it will be visible in the published service. This name can be different from Name.
- *HTTP request parameter required* - If checked, the parameter is mandatory and an error will be reported if it's omitted.
- *Pass HTTP request body* - If checked, the request body is set to dictionary entry as readable channel.
- *Pass value as graph parameter* - If checked, the property value will be passed to the job also as a parameter (${ParameterName}, where ParameterName is equal to Name). This lets you use the parameter anywhere in the job definition (not just places that support Dictionary). However, parameters are evaluated during job initialization. Thus, such a job cannot be pooled which decreases performance for high frequency repetitive calls to the service. In this case, consider redesigning the transformation to use Dictionary instead, allowing for pooling.
- *Default parameter value* - The default value applied in case the parameter is omitted in the launch request.

Launch services authentication

If you are using launch services, you have two ways how to be logged in: using form-based authentication of Server console or HTTP basic authentication of Launch services.

The form-based authentication of Server console enables user to create or modify Launch services. If you are logged in this way, you act as an administrator of Launch services.

To insert data into the Launch service form you should be logged in using HTTP basic authentication. Follow the link to the Launch service form and web browser will request your credentials. If you are logged in usin HTTP-basic authentication you act as an user of Launch services forms.

Sending the Data to Launch Service

A launch request can be sent via HTTP GET or POST methods. A launch request is simply a URL which contains the values of all parameters that should be passed to the job. The request URL is composed of several parts:

(You can use a Launch Services test page, accessible from the login screen, to test drive Launch Services.)

```
[Clover Context]/launch/[Configuration name]?[Parameters]
```

- `[Clover Context]` is the URL to the context in which the CloverETL Server is running. Usually this is the full URL to the CloverETL Server (for example, for CloverETL Demo Server this would be http://server-demo.cloveretl.com:8080/clover).
- `[Configuration name]` is the name of the launch configuration specified when the configuration was created. In our example, this would be set to “mountains” (case-sensitive).
- `[Parameters]` is the list of parameters the configuration requires as a query string. It’s a URL-encoded [RFC 1738] list of name=value pairs separated by the "&" character.

Based on the above, the full URL of a launch request for our example with mountains may look like this: http://server-demo.cloveretl.com:8080/clover/launch/NewMountains?heightMin=4000. In the request above, the value of heightMin property is set to 4000.

Results of the Graph Execution

After the job terminates, the results are sent back to the HTTP client as content of an HTTP response.

Output parameters are defined in the job’s Dictionary. Every Dictionary entry marked as “Output” is sent back as part of the response.

Depending on the number of output parameters, the following output is sent to the HTTP client:

- *No output parameters* - Only a summary page is returned. The page contains details such as: when the job was started, when it finished, the user name, and so on. The format of the summary page cannot be customized.
- *One output parameter* - In this case, the output is sent to the client as in the body of the HTTP response with its MIME content type defined by the property type in Dictionary.
- *Multiple output parameters* - In this case, each output parameter is sent to the HTTP client as part of multipart HTTP response. The content type of the response is either multipart/related or multipart/x-mixed-replace, depending on the HTTP client (the client detection is fully automatic). The multipart/related type is used for browsers based on Microsoft Internet Explorer and the multipart/x-mixed-replace is sent to browsers based on Gecko or Webkit.

Launch requests are recorded in the log files in the directory specified by the `launch.log.dir` property in the CloverETL Server configuration. For each launch configuration, one log file named [Configuration name]#[Launch ID].log is created. For each launch request, this file will contain only one line with following tab-delimited fields:

(If the property `launch.log.dir` is not specified, log files are created in the temp directory `[java.io.tmpdir]/cloverlog/launch` where "java.io.tmpdir" is system property)

- *Launch start time*
- *Launch end time*
- *Logged-in user name*
- *Run ID*
- *Execution status* FINISHED_OK, ERROR or ABORTED
- *IP Address* of the client
- *User agent* of the HTTP client
- *Query string* passed to the Launch Service (full list of parameters of the current launch)

In the case that the configuration is not valid, the same launch details are saved into the _no_launch_config.log file in the same directory. All unauthenticated requests are saved to the same file as well.

CloverETL Server API Extensibility

The CloverETL Server implements extensibility of its APIs, so the Server may expose additional features with a custom API.

There are two possibilities: The Groovy code API and the OSGi plugin.

Groovy Code API

Since 3.3

The CloverETL Server Groovy Code API allows clients to execute Groovy code stored on the Server by an HTTP request. Executed code has access to the ServerFacade, instance HTTP request and HTTP response, so it's possible to implement a custom CloverETL Server API in the Groovy code.

To execute the code, call URL:

```
http://[host]:[port]/clover/groovy/[sandboxCode]/[pathToGroovyCodeFile]
```

Protocol HTTP can be changed to secured HTTPS according to the web server configuration.

The Server uses Basic or Digest authentication according to the configuration. So, the user must be authorized and must have permission to execute in the specified sandbox and permission to call "Groovy Code API".

Important

Note that permission to call "Groovy Code API" (and edit them) is a very strong permission, since the Groovy Code can basically do the same as Java code and it runs as the same system process as a whole application container.

Variables accessible in the Groovy code

By default, these variables are accessible to the Groovy code

Table 23.15. Variables accessible in groovy code

type	key	description
javax.servlet.http.HttpServletRequest	request	Instance of HTTP request, which triggered the code.
javax.servlet.http.HttpServletResponse	response	Instance of HTTP response, which will be sent to the client when the script finishes.
javax.servlet.http.HttpSession	session	Instance of HTTP session.
javax.servlet.ServletConfig	servletConfig	instance of ServletConfig
javax.servlet.ServletContext	servletContext	instance of ServletContext
com.cloveretl.server.api.ServerFacade	serverFacade	Instance of serverFacade usable for calling CloverETL Server core features. WAR file contains JavaDoc of facade API and it is accessible on URL: http://host:port/clover/javadoc/index.html
String	sessionToken	sessionToken, needed for calling serverFacade methods

Code examples

Code may return string that will be returned as content of the HTTP response – or it may construct the output itself to the output Writer

The following script writes its own output and doesn't return anything, so the underlying servlet doesn't modify the output at all. The advantage of this approach is that the output may be constructed on the fly and sent to the client continuously. However, when the output stream (or writer) is opened, the servlet won't send any error description in case of any error during the script processing.

```
response.getWriter().write("write anything to the output");
```

The following script returns String, so the underlying servlet puts the string to the output. The advantage of this approach is that in case of any error during code processing, the servlet returns a full stacktrace, so the script may be fixed. However, the constructed output may consume some memory.

```
String output = "write anything to the output";
return output;
```

The following script is little more complex. It returns XML with a list of all configured schedules. You need to have permission to list the schedules.

```
// uses variables: response, sessionToken, serverFacade
import java.io.*;
import java.util.*;
import javax.xml.bind.*;
import com.cloveretl.server.facade.api.*;
import com.cloveretl.server.persistent.*;
import com.cloveretl.server.persistent.jaxb.*;

JAXBContext jc = JAXBContext.newInstance( "com.cloveretl.server.persistent:com.cloveretl.server.persistent.jaxb" )
Marshaller m = jc.createMarshaller();
m.setProperty(Marshaller.JAXB_ENCODING, "UTF-8");
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
m.setProperty(Marshaller.JAXB_SCHEMA_LOCATION, "/clover/schemas/cs.xsd");

Response<List<Schedule>> list = serverFacade.findScheduleList(sessionToken, null);
SchedulesList xmlList = new SchedulesList();
xmlList.setSchedulesList(list.getBean());
m.marshal(xmlList, response.getWriter());
```

Embedded OSGi Framework

Since 3.0

The CloverETL Server includes an embedded OSGi framework which allows for implementation of a "plugin" (an OSGi bundle). It can add a new API operation or even extend the Server Console UI. It is independent of the standard clover.war.

CloverETL itself isn't based on OSGi technology, OSGi is used only optionally for extending server APIs. OSGi framework is completely disabled by default and is enabled only when the property "plugins.path" is set as described below.

Embedded OSGi framework Equinox uses some invasive techniques, so it may interfere with other applications or even some CloverETL features. Generic recommendation is to use Groovy API explained above instead, however

there are still use cases when the OSGi plugin is better choice. E.g. custom API has to use different libraries then the ones on the server classpath. Whereas groovy uses the same classpath as CloverETL, OSGi plugin has its own isolated classpath. So in such case, when the OSGi plugin has to be used, CloverETL should be deployed in the environment which is as simple as possible, e.g. Clover is the only app deployed on the container, which should be also lightweight - Tomcat or Jetty. Such deployment would minimalize chance of possible conflicts.

Examples of interferences with embedded OSGi framework (only when framework is explicitly configured and initialized)

- OSGi framework causes malfunction of CloverETL components WebServiceClient and EmailReader on IBM WebSphere
- OSGi framework causes failure of CloverETL startup on WebLogic 10.3.5 running on Oracle JRockit JVM
- OSGi framework can't initialize itself properly on WebLogic 12
- OSGi framework can't initialize itself on JBoss EAP 6

Plugin possibilities

Basically, the plugin may work as new server API similarly as Launch Services, HTTP API, or WebServices API. It may just be simple JSP, HttpServlet or complex SOAP Web Services. If the plugin contains some HTTP service, it’s registered to listen on a specified URL during the startup and incoming HTTP requests are "bridged" from the web container to the plugin. The plugin itself has access to the internal CloverETL Server interface called "ServerFacade". ServerFacade offers methods for executing graphs, obtaining graph status and executions history, and manipulation with scheduling, listeners, configuration, and many more. So the API may be customized according to the needs of a specific deployment.

Deploying an OSGi bundle

There are two CloverETL Server configuration properties related to the OSGi framework:

- plugins.path - Absolute path to the directory containing all your plugins (JAR files).
- plugins.autostart - It is a comma-separated plugin names list. These plugins will be started during the server startup. Theoretically, the OSGi framework can start the OSGi bundle on demand; however, it is unreliable when the servlet bridge to the servlet container is used, so we strongly recommended naming all your plugins.

To deploy your plugin: set the two config properties, copy the plugin to the directory specified by "plugins.path" and restart the server.

Chapter 24. Recommendations for transformations developers

Add external libraries to app-server classpath

Connections (JDBC/JMS) may require third-party libraries. We strongly recommended adding these libraries to the app-server classpath.

CloverETL allows you to specify these libraries directly in a graph definition so that CloverETL can load these libraries dynamically. However, external libraries may cause memory leak, resulting in "java.lang.OutOfMemoryError: PermGen space" in this case.

In addition, app-servers should have the JMS API on their classpath – and the third-party libraries often bundle this API as well. So it may result in classloading conflicts if these libraries are not loaded by the same classloader.

Another graphs executed by RunGraph component may be executed only in the same JVM instance

In the server environment, all graphs are executed in the same VM instance. The attribute "same instance" of the RunGraph component cannot be set to false.

Chapter 25. Extensibility - CloverETL Engine Plugins

Since 3.1.2

The CloverETL Server can use external engine plugins loaded from a specified source. The source is specified by config property "engine.plugins.additional.src"

See details about the possibilities with CloverETL configuration in *Part III, "*Configuration*" (p. 40)*

This property must be the absolute path to the directory or zip file with additional CloverETL engine plugins. Both the directory and zip must contain a subdirectory for each plugin. These plugins are not a substitute for plugins packed in WAR. Changes in the directory or the ZIP file apply only when the server is restarted.

Each plugin has its own class-loader that uses a parent-first strategy by default. The parent of plugins' classloaders is web-app classloader (content of [WAR]/WEB-INF/lib). If the plugin uses any third-party libraries, there may be some conflict with libraries on parent-classloaders classpath. These are common exceptions/errors suggesting that there is something wrong with classloading:

- java.lang.ClassCastException
- java.lang.ClassNotFoundException
- java.lang.NoClassDefFoundError
- java.lang.LinkageError

There are a couple of ways you can get rid of such conflicts:

- Remove your conflicting third-party libraries and use libraries on parent classloaders (web-app or app-server classloaders)
- Use a different class-loading strategy for your plugin.
  - In the plugin descriptor plugin.xml, set attribute greedyClassLoader="true" in the element "plugin"
  - It means that the plugin classloader will use a self-first strategy
- Set an inverse class-loading strategy for selected Java packages.
  - In the plugin descriptor plugin.xml, set attribute "excludedPackages" in the element "plugin".
  - It's a comma-separated list of package prefixes – like this, for example: excludedPackages="some.java.package,some.another.package"
  - In the previous example, all classes from "some.java.package", "some.another.package" and all their sub-packages would be loaded with the inverse loading strategy, then the rest of classes on the plugins classpath.

The suggestions above may be combined. It's not easy to find the best solution for these conflicts and it may depend on the libraries on app-server classpath.

For more convenient debugging, it’s useful to set TRACE log level for related class-loaders.

```
<logger name="org.jetel.util.classloader.GreedyURLClassLoader">
 <level value="trace"/>
</logger>
<logger name="org.jetel.plugin.PluginClassLoader">
 <level value="trace"/>
</logger>
```

See Chapter 11, Logging (p. 74) for details about overriding a server log4j configuration.

Chapter 26. Troubleshooting

Graph hangs and is un-killable

Graph can sometimes hang and be un-killable if some network connection in it hangs. This can be improved by setting a shorter tcp-keepalive so that the connection times out earlier. The default value on Linux is 2 hours (7,200 seconds). You can set it to 10 minutes (600 seconds).

See http://tldp.org/HOWTO/TCP-Keepalive-HOWTO/usingkeepalive.html on tcp-timeout in Linux.

The file descriptor can be closed manually using `gdb`. See http://stackoverflow.com/questions/5987820/how-to-close-file-descriptor-via-linux-shell-command/12058001#12058001.

Part VI. Cluster

Chapter 27. Clustering features

There are two common Cluster features: high availability and scalability. Both are implemented by the CloverETL Server on different levels. This section should clarify the basics of CloverETL Clustering.

The CloverETL Server only works in the Cluster if your license allows it.

High Availability

CloverETL Server does not recognize any differences between cluster nodes. Thus, there are no "master" or "slave" nodes meaning all nodes can be virtually equal. There is no single point of failure (SPOF) in the CloverETL cluster itself, however SPOFs may be in the input data or some other external element.

Clustering offers high availability (HA) for all features accessible through HTTP, for event listeners and scheduling. Regarding the HTTP accessbile features: it includes sandbox browsing, modification of services configuration (scheduling, launch services, listeners) and primarily job executions. Any cluster node may accept incoming HTTP requests and process them itself or delegate it to another node.

Since all nodes are typically equal, almost all requests may be processed by any cluster node:

- All job files, metadata files, etc. are located in shared sandboxes. Thus all nodes have access to them. A shared filesystem may be a SPOF, thus it is recommended to use a replicated filesystem instead.
- The database is shared by all cluster nodes. Again, a shared DB might be a SPOF, however it may be clustered as well.

But there is still a possibility, that a node cannot process a request by itself. In such cases, it completely and transparently delegates the request to a node which can process the request.

These are the requests which are limited to one (or more) specific node(s):

- a request for the content of a partitioned or local sandbox. These sandboxes aren't shared among all cluster nodes. Please note that this request may come to any cluster node which then delegates it transparently to a target node, however, this target node must be up and running.
- A job is configured to use a partitioned or local sandbox. These jobs need nodes which have a physical access to the required sandboxes.
- A job has allocation specified by specific cluster nodes. Concept of "allocation" is described in the following sections.

Thus an inaccessible cluster node may cause a failure of the request, so if it's possible, it's better to avoid using specific cluster nodes or resources accessible only by specific cluster node.

CloverETL itself implements a load balancer for executing jobs. So a job which isn't configured for some specific node(s) may be executed anywhere in the cluster and the CloverETL load balancer decides, according to the request and current load, which node will process the job. All this is done transparently for the client side.

To achieve HA, it is recommended to use an independent HTTP load balancer. Independent HTTP load balancers allow transparent fail-overs for HTTP requests. They send requests to the nodes which are running.

Scalability

There are two independent levels of scalability implemented. Scalability of transformation requests (and any HTTP requests) and data scalability (parallel data processing).

Both of these "scalability levels" are "horizontal". Horizontal scalability means adding nodes to the cluster, whereas vertical scalability means adding resources to a single node. Vertical scalability is supported natively by the CloverETL engine and it is not described here.

Transformation Requests

Basically, the more nodes we have in the cluster, the more transformation requests (or HTTP requests in general) we can process at one time. This type of scalability is the CloverETL server's ability to support a growing number of clients. This feature is closely related to the use of an HTTP load balancer which is mentioned in the previous section.

Parallel Data Processing

This type of scalability is currently available only for ETL graphs. Jobflow and Profiler jobs can't run in parallel.

When a transformation is processed in parallel, the whole graph (or its parts) runs in parallel on multiple cluster nodes having each node process just a part of the data.

So the more nodes we have in the cluster, the more data can be processed in the specified time.

The data may be split (partitioned) before the graph execution or by the graph itself on the fly. The resulting data may be stored in partitions or gathered and stored as one group of data.

The curve of scalability may differ according to the type of transformation. It may be almost linear, which is almost always ideal, except when there is a single data source which cannot be read by multiple readers in parallel limiting the speed of further data transformation. In such cases it is not beneficial to have parallel data processing since it would actually wait for input data.

ETL Graph Allocation

Each ETL graph executed in cluster environment is automatically subjected to transformation analysis. The main goal of this analysis is to find so called ETL graph **allocation**. The graph allocation is set of instructions for cluster environment how the transformation should be executed. For better understanding how the parallel data processing works, it is necessary to get deeper information about the graph analysis and resulted allocation.

First of all, analysis needs to find allocation for each individual component. The component allocation is set of cluster nodes where the component should be running. There are several ways how the component allocation can be specified, see following section of the documentation. But important information for now is, that a component can be requested to run in multiple instances - that is necessary for parallel data processing. Next step of analysis is to find optimal graph decomposition to ensure all component allocation will be satisfied and tries to minimise number of remote edges between graph instances.

Resulted analysis says how many instances (workers) of the graph needs to be executed, on which cluster nodes these instances will be running and which components will be present in the instances. In other words, one executed graph can be running in many instances, each instance can be processed on arbitrary cluster node and moreover each instance contains only convenient components.

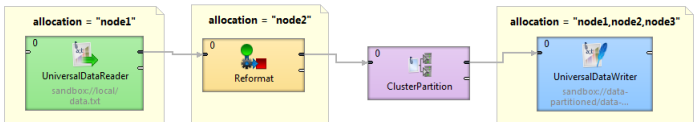


Figure 27.1. Component allocations example

This figure shows sample graph with few components with various component allocations. First component UniversalDataReader requests to be executed on node1, following Reformat component should be running on cluster node2, the ClusterPartition component is special component which makes possible to change cardinality of allocation of two interconnected components (detailed description of cluster partitioning and gathering follows this section). The last component UniversalDataWriter requires to be executed right on three cluster nodes node1, node2 and node3. Visualisation of transformation analysis shows the following figure. Three workers (graphs) will be executed, each on different cluster node (which is not necessary, even multiple workers can be associated with a single node). Worker on cluster node1 contains only UniversalDataReader and first of three instances

of UniversalDataWriter component. Both components are connected by remote edges with components, which are running on node2. The worker running on node3 contains only UniversalDataWriter fed by data remotely transfered from ClusterPartitioner running on node2.

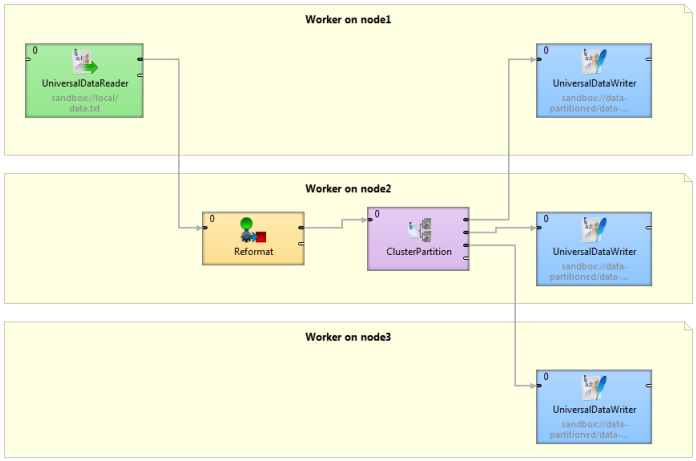


Figure 27.2. Graph decomposition based on component allocations

Component Allocation

Allocation of a single component can be derived in several ways (list is ordered according priority):

- **Explicit definition** - all components have common attribute **Allocation**. CloverETL Designer allows user to use convenient dialog.

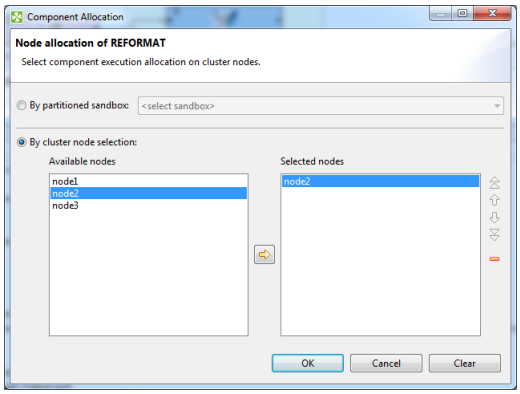


Figure 27.3. Component allocation dialog

  Three different approaches are available for explicit allocation definition:

  - Allocation based on number of workers - component will be executed in requested instances on some cluster nodes, which are preferred by CloverETL Cluster. Server can use build-in loadbalancing algorithm to ensure fastest data processing.

  - Allocation based on reference on a partitioned sandbox - component allocation corresponds with locations of given partitioned sandbox. Each partitioned sandbox has a list of locations, each bound to specific cluster node. Thus allocation would be equivalent to list of locations. See "Partitioned sandbox" in Partitioned and Local Sandboxes (p. 186) for details.

  - allocation defined by list of cluster node identifiers (single cluster node can be used more times)

- **Reference to a partitioned sandbox** UniversalDataReader, UniversalDataWriter and ParallelReader components derives theirs allocation from fileURL attribute. In case the URL refers to a file in a partitioned sandbox, the component allocation is automatically derived from locations of the partitioned sandbox. So in

case you manipulate with one of these components with a file in partitioned sandbox suitable allocation is used automatically.

- **Adoption from neighbour components** By default, allocation is inherited from neighbour components. Components on the left side have higher priority. Cluster partitioners and cluster gathers are nature bounds for recursive allocation inheritance.

Partitioning/gathering Data

As mentioned before, data may be partitioned and gathered in multiple ways. It may be prepared before the graph is executed or it may be partitioned on the fly.

Partitioning/gathering "on the fly"

There are six special components to consider: ClusterPartition, ClusterLoadBalancingPartition, ClusterSimpleCopy, ClusterSimpleGather, ClusterMerge and ClusterRepartition. All work similarly to their non-cluster variation. But their splitting or gathering nature is used to change data flow allocation, so they may be used to change distribution of the data among workers.

ClusterPartition and **ClusterLoadBalancingPartition** work similar to a common partitioner, change the data allocation from 1 to N. Component preceding the ClusterPartitioner run on just one node, whereas component behind the ClusterPartitioner run in parallel according to node allocation. **ClusterSimpleCopy** component can be use in similar locations, this component does not distribute the data records, but copies them to all output workers.

ClusterGather and **ClusterMerge** work in the opposite way. They change the data allocation from N to 1. Component preceding the gather/merge run in parallel while component behind the gather run on just one node.

Partitioning/gathering data by external tools

Partitioning data on the fly may in some cases be an unnecessary bottleneck. Splitting data using low-level tools can be much better for scalability. The optimal case being, that each running worker reads data from an independent data source. Thus there does not have to be a ClusterPartitioner component and the graph runs in parallel from the beginning.

Or the whole graph may run in parallel, however the results would be partitioned.

Node Allocation Limitations

As described above, each component may it's own node allocation specified which may result in some conflicts.

- *Node allocation of neighbouring components must have the same cardinality* So it doesn't have to be the same allocation, but the cardinality must be the same. E.g. There is an ETL graph with 2 components: DataGenerator and Trash. DataGenerator allocated on nodeA sending data to Trash allocated on nodeB works fine. DataGenerator allocated on nodeA sending data to Trash allocated on nodeA and nodeB fails.
- *Node allocation behind the ClusterGather and ClusterMerge must have cardinality 1* So it may be any allocation, but the cardinality must be just 1.
- *Node allocation of components in front of the ClusterPartition, ClusterLoadBalancingPartition and ClusterSimpleCopy must have cardinality 1*

Partitioned and Local Sandboxes

Partitioned and local sandboxes were mentioned in previous sections. These new sandbox types were introduced in version 3.0 and they are vital for parallel data processing.

Together with shared sandboxes, we have three sandbox types in total.

Shared sandbox

This type of sandbox must be used for all data which is supposed to be accessible on all cluster nodes. This includes all graphs, jobflows, metadata, connections, classes and input/output data for graphs which should support HA, as

described above. All shared sandboxes reside in the directory, which must be properly shared among all cluster nodes. You can use suitable sharing/replicating tool according to the operating system and filesystem.

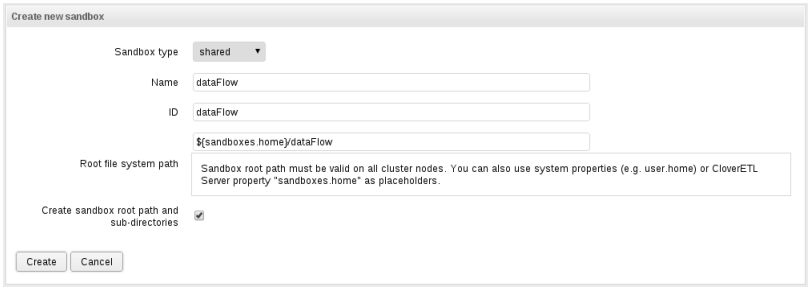


Figure 27.4. Dialog form for creating new shared sandbox

As you can see in the screenshot above, you can specify the root path on the filesystem and you can use placeholders or absolute path. Placeholders available are environment variables, system properties or CloverETL Server config property indended for this use `sandboxes.home`. Default path is set as `[user.data.home]/CloverETL/sandboxes/[sandboxID]` where the `sandboxID` is ID specified by the user. The `user.data.home` placeholder refers to the home directory of the user running the Java Virtual Machine process (`/home` subdirectory on Unix-like OS); it is determined as first writable directory selected from following values:

- `USERPROFILE` environment variable on Windows OS
- `user.home` system property (user home directory)
- `user.dir` system property (JVM process working directory)
- `java.io.tmpdir` system property (JVM process temporary directory)

Note that the path must be valid on all cluster nodes. Not just nodes currently connected to the cluster, but also on the nodes that may be connected later. Thus when the placeholders are resolved on the node, the path must exist on the node and it must be readable/writeable for the JVM process.

Local sandbox

This sandbox type is intended for data, which is accessible only by certain cluster nodes. It may include massive input/output files. The purpose being, that any cluster node may access content of this type of sandbox, but only one has local (fast) access and this node must be up and running to provide data. The graph may use resources from multiple sandboxes which are physically stored on different nodes since cluster nodes are able to create network streams transparently as if the resource were a local file. See Using a Sandbox Resource as a Component Data Source (p. 188) for details.

Do not use local sandbox for common project data (graphs, metadata, connections, lookups, properties files, etc.). It would cause odd behaviour. Use shared sandboxes instead.

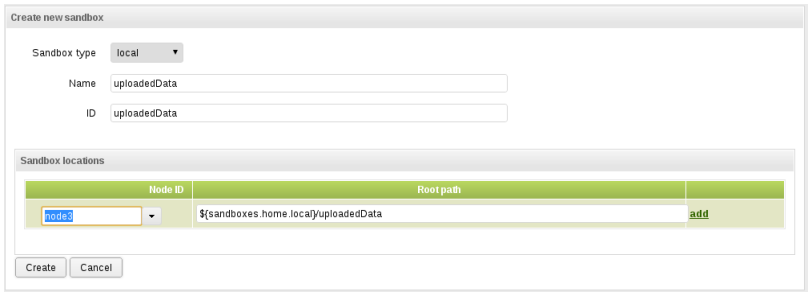


Figure 27.5. Dialog form for creating new local sandbox

Sandbox location path is pre-filled with placeholder `sandboxes.home.local` which by default points to the `[user.data.home]/CloverETL/sandboxes-local`. The placeholder can be configured as any other CloverETL configuration property.

Partitioned sandbox

This type of sandbox is actually an abstract wrapper for a couple of physical locations existing typically on different cluster nodes. However, there may be multiple locations on the same node. A partitioned sandbox has two purposes which are both closely related to parallel data processing.

1. **node allocation** specification - locations of a partitioned sandbox define the workers which will run the graph or its parts. So each physical location will cause a single worker to run. This worker does not have to actually store any data to "its" location. It is just a way to tell the CloverETL Server: "execute this part of ETL graph in parallel on these nodes"

2. **storage for part of the data** during parallel data processing. Each physical location contains only part of the data. In a typical use, we have input data split in more input files, so we put each file into a different location and each worker processes its own file.

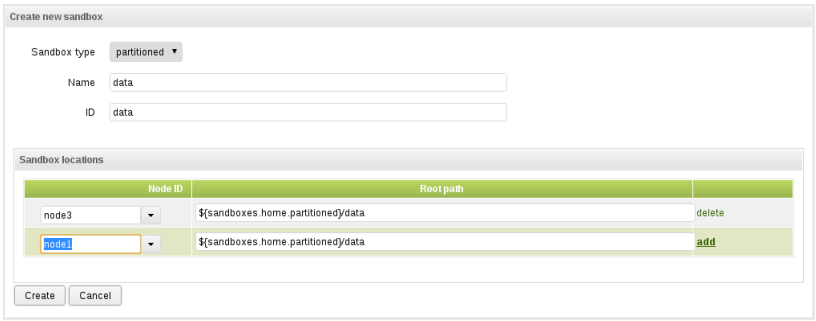


Figure 27.6. Dialog form for creating new local sandbox

As you can see on the screenshot above, for a partitioned sandbox, you can specify one or more physical locations on different cluster nodes.

Sandbox location path is pre-filled with placeholder `sandboxes.home.partitioned` which by default points to the `[user.data.home]/CloverETL/sandboxes-paritioned`. Anyway the config property `sandboxes.home.partitioned` may be configured as any other CloverETL Server configuration property. Note that directory must be readable/writeable for the user running JVM process.

Do not use partitioned sandbox for common project data (graphs, metadata, connections, lookups, properties files, etc.). It would cause odd behavior. Use shared sandboxes instead.

Using a Sandbox Resource as a Component Data Source

A sandbox resource, whether it is a shared, local or partitioned sandbox (or ordinary sandbox on standalone server), is specified in the graph under the fileURL attributes as a so called sandbox URL like this:

```
sandbox://data/path/to/file/file.dat
```

where "data" is a code for sandbox and "path/to/file/file.dat" is the path to the resource from the sandbox root. URL is evaluated by CloverETL Server during job execution and a component (reader or writer) obtains the opened stream from the server. This may be a stream to a local file or to some other remote resource. Thus, a job does not have to run on the node which has local access to the resource. There may be more sandbox resources used in the job and each of them may be on a different node.

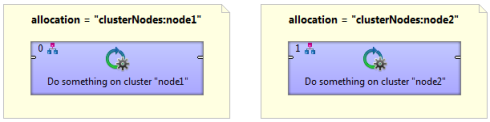
The sandbox URL has a specific use for parallel data processing. When the sandbox URL with the resource in a *partitioned sandbox* is used, that part of the graph/phase runs in parallel, according to the node allocation specified by the list of partitioned sandbox locations. Thus, each worker has it is own local sandbox resource. CloverETL Server evaluates the sandbox URL on each worker and provides an open stream to a local resource to the component.

The sandbox URL may be used on standalone server as well. It is excellent choice when graph references some resources from different sandboxes. It may be metadata, lookup definition or input/output data. Of course, referenced sandbox must be accessible for the user who executes the graph.

Graph allocation examples

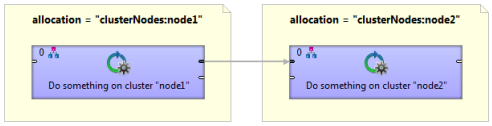
Basic component allocation

This example shows two component graph, where allocation ensures the first component will be executed on cluster node1 and the second component will be executed on cluster node2.



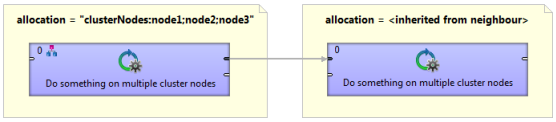
Basic component allocation with remote data transfer

Two components connected with an edge can have different allocation. The first is executed on node1 and the second is executed on node2. Cluster environment automatically ensures remote data records transfer.



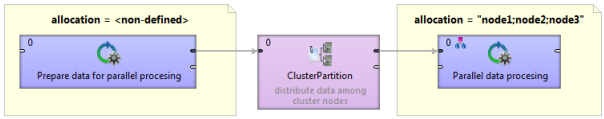
Multiple execution

Graph with multiple node allocation is executed in parallel. In this example both components have same allocation, so three identical transformation will be executed on cluster node1, node2 and node3.



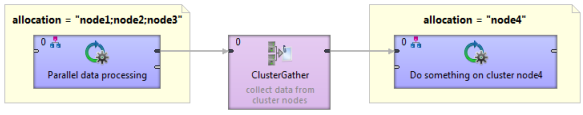
Cluster data partitioning

Graph with two allocations. First component has single node allocation, which is not specified and is automatically derived to ensure minimal number of remote edges. The ClusterPartition component distribute records for further data processing on cluster node1, node2 and node3.



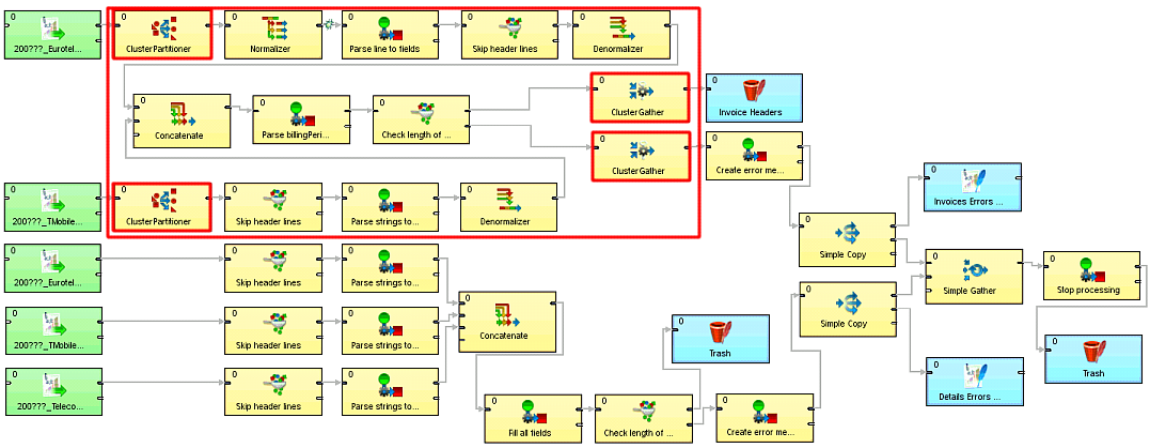
Cluster data gathering

Graph with two allocations. Resulted data records of parallel data processing in the first component are collected in ClusterGather component and passed to cluster node4 for further single node processing.



Example of Distributed Execution

The following diagram shows a transformation graph used for parsing invoices generated by a few cell phone network providers in Czech Republic.



The size of these input files may be up to a few gigabytes, so it is very beneficial to design the graph to work in the cluster environment.

Details of the Example Transformation Design

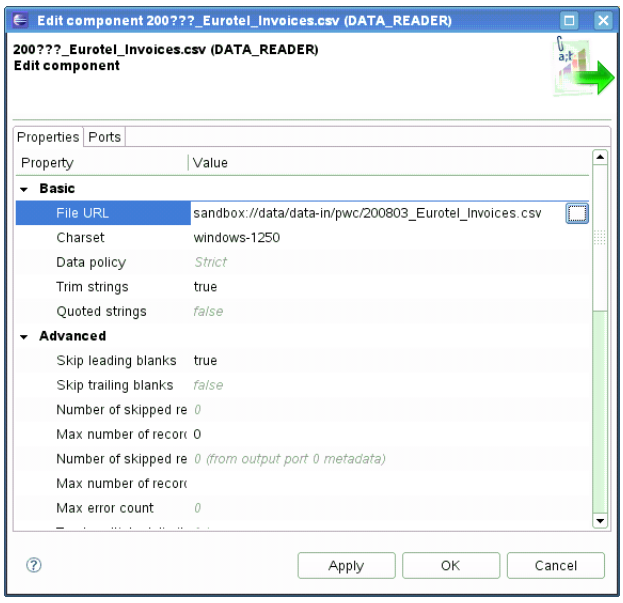
Please note there are four cluster components in the graph and these components define a point of change "node allocation", so the part of the graph demarcated by these components is highlighted by the red rectangle. Allocation of these component should be performed in parallel. This means that the components inside the dotted rectangle should have convenient allocation. The rest of the graph runs just on single node.

Specification of "node allocation"

There are 2 node allocations used in the graph:

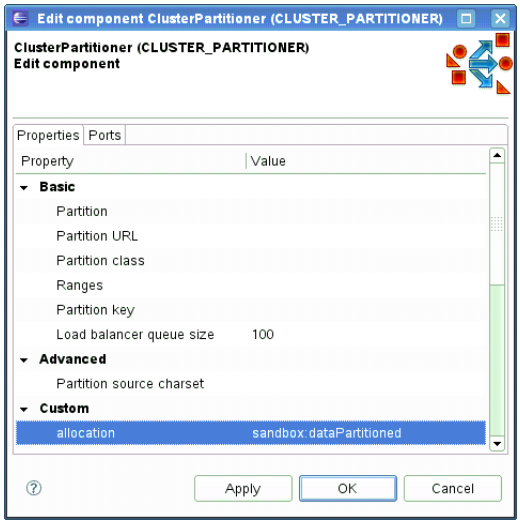
- node allocation for components running in parallel (demarcated by the four cluster components)
- node allocation for outer part of the graph which run on a single node

The single node is specified by the sandbox code used in the URLs of input data. The following dialog shows the File URL value: "sandbox://data/path-to-csv-file", where "data" is the ID of the server sandbox containing the specified file. And it is the "data" *local* sandbox which defines the single node.



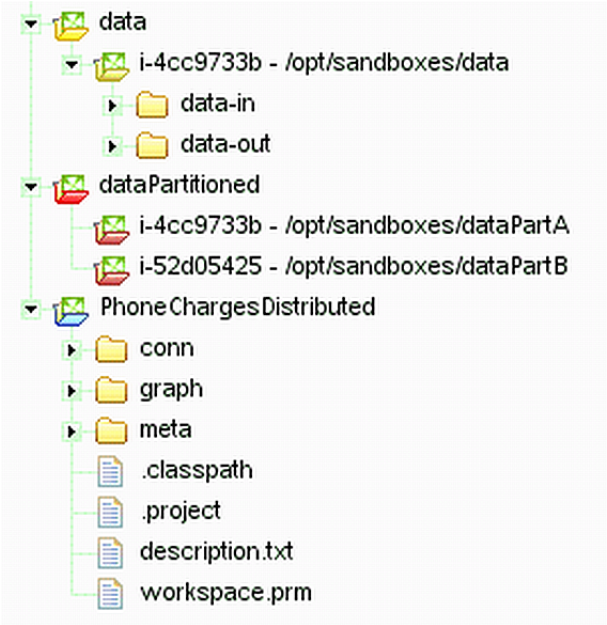
The part of the graph demarcated by the four cluster components may have specified its allocation by the file URL attribute as well, but this part does not work with files at all, so there is no file URL. Thus, we will use the "node allocation" attribute. Since components may adopt the allocation from their neighbours, it is sufficient to set it only for one component.

Again, "dataPartitioned" in the following dialog is the sandbox ID.



Let's investigate our sandboxes. This project requires 3 sandboxes: "data", "dataPartitioned" and "PhoneChargesDistributed".

- data
  - contains input and output data
  - local sandbox (yellow folder), so it has only one physical location
  - accessible only on node "i-4cc9733b" in the specified path
- dataPartitioned
  - partitioned sandbox (red folder), so it has a list of physical locations on different nodes
  - does not contain any data and since the graph does not read or write to this sandbox, it is used only for the definition of "nodes allocation"
  - on the following figure, allocation is configured for two cluster nodes
- PhoneChargesDistributed
  - common sandbox containing the graph file, metadata, and connections
  - shared sandbox (blue folder), so all cluster nodes have access to the same files



If the graph was executed with the sandbox configuration of the previous figure, the node allocation would be:

- components which run only on single node, will run only on the "i-4cc9733b" node according to the "data" sandbox location.
- components with allocation according to the "dataPartitioned" sandbox will run on nodes "i-4cc9733b" and "i-52d05425".

Scalability of the Example Transformation

The example transformation has been tested in the Amazon Cloud environment with the following conditions for all executions:

- the same master node
- the same input data: 1,2 GB of input data, 27 million records
- three executions for each "node allocation"
- "node allocation" changed between every 2 executions
- all nodes has been of "c1.medium" type

We tested "node allocation" cardinality from 1 single node, all the way up to 8 nodes.

The following figure shows the functional dependence of run-time on the number of nodes in the cluster:

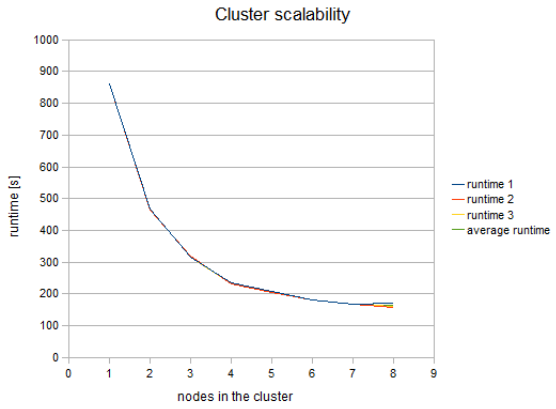


Figure 27.7. Cluster Scalability

The following figure shows the dependency of "speedup factor" on the number of nodes in the cluster. The speedup factor is the ratio of the average runtime with one cluster node and the average runtime with x cluster nodes. Thus:

```
speedupFactor = avgRuntime(1 node) / avgRuntime(x nodes)
```

We can see, that the results are favourable up to 4 nodes. Each additional node still improves cluster performance, however the effect of the improvement decreases. Nine or more nodes in the cluster may even have a negative effect because their benefit for performance may be lost in the overhead with the management of these nodes.

These results are specific for each transformation, there may be a transformation with much a better or possibly worse function curve.

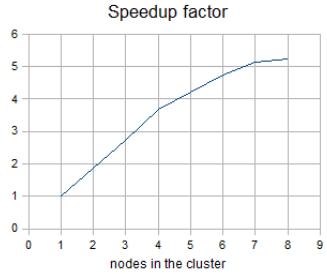


Figure 27.8. Speedup factor

Table of measured runtimes:

nodes	runtime 1 [s]	runtime 2 [s]	runtime 3 [s]	average runtime [s]	speedup factor
1	861	861	861	861	1
2	467	465	466	466	1.85
3	317	319	314	316.67	2.72
4	236	233	233	234	3.68
5	208	204	204	205.33	4.19
6	181	182	182	181.67	4.74
7	168	168	168	168	5.13
8	172	159	162	164.33	5.24

Chapter 28. Cluster configuration

Cluster can work properly only if each node is properly configured. Clustering must be enabled, nodeID must be unique on each node, all nodes must have access to shared DB (direct connection or proxied by another cluster node) and shared sandboxes, and all properties for inter-node cooperation must be set according to network environment.

Properties and possible configuration are the following:

Mandatory properties

Besides mandatory cluster properties, you need to set other necessary properties which are not specifically related to the cluster environment. Database connection must be also configured, however besides direct connection it's alternatively possible to configure proxing using another cluster node/nodes. See property cluster.datasource.type (p. 197) for details.

Table 28.1. Mandatory properties - these properties must be properly set on each node of the cluster

property	type	default
cluster.enabled	boolean	false
description:	switch whether server is connected to the cluster or not	
cluster.node.id	String	node01
description:	each cluster node must have unique ID	
cluster.jgroups.bind_address	String, IP address	127.0.0.1
description:	IP address of ethernet interface, which is used for communication with another cluster nodes. Necessary for inter-node messaging.	
cluster.jgroups.start_port	int, port	7800
description:	Port where jGroups server listens for inter-node messages.	
cluster.http.url	String, URL	http://localhost:8080/clover
description:	URL of the CloverETL cluster node. It must be HTTP/HTTPS URL to the root of web application, thus typicaly it would be "http://[hostname]:[port]/clover". Primarily it's used for synchronous inter-node communication from other cluster nodes. It's recommended to use a fully qualified hostname or IP address, so it's accessible from client browser or CloverETL Designer.	

Following property must be set only when the node uses "remote" DB datasource (See property cluster.datasource.type (p. 197) for details). When the node doesn't have the direct DB connection, it can't interchange some config data with other nodes, so it's necesssary to configure them explicitly.

Table 28.2. Mandatory property for remote DB datasource access

property	type	default
cluster.jgroups.tcpping.initial_hosts	String, in format: "IPaddress1[port1],IPaddress2[port2]"	127.0.0.1[7800]
description:	List of IP addresses (with ports) where we expect running and listening nodes. It is related to another nodes "bind_address" and "start_port" properties. I.e. like this: bind_address1[start_port1],bind_address2[start_port2],... It is not	

property	type	default
	necessary to list all nodes of the cluster, but at least one of listed host:port must be running. Necessary for inter-node messaging.	

Optional properties

Table 28.3. Optional properties - these properties aren't vital for cluster configuration - default values are sufficient

property	type	default	description
cluster.jgroups.external_address	String, IP address		IP address of the cluster node. Configure this only if the cluster nodes are on the different sub-nets, so IP address of the network interface isn't directly accessible from the other cluster nodes.
cluster.jgroups.external_port	int, port		Port for asynchronous messaging. Configure this only if the cluster nodes are on the different sub-nets, and port opened on the IP address is different then port opened on the node's network interface IP address.
sandboxes.home.partitioned	String	${user.data.home}/CloverETL/sandboxes-partitioned	This property is indended to be used as placeholder in the location path of partitioned sandboxes. So the sandbox path is specified with the placeholder and it's resolved to the real path just before it's used. The default value uses configuration property "user.data.home" which points to the home directory of the user which runs the JVM process. Directory depends on the OS. On uni-like systems it's typically /home/[username]
sandboxes.home.local	String	${user.data.home}/CloverETL/sandboxes-local	This property is indended to be used as placeholder in the location path of local sandboxes. So the sandbox path is specified with the placeholder and it's resolved to the real path just before it's used. The default value uses configuration property "user.data.home" which points to the home directory of the user which runs the JVM process. Directory depends on the OS. On Unix-like systems it's typically /home/[username]

property	type	default	description
cluster.shared_sandboxes_path	String		This property is deprecated. This property still works but it's used only when shared sandbox doesn't have it's own path specified. It's just for backward compatibility and it's not recommended for new deployments. Since 3.5 it's recommended to specify sandbox path explicitly and use "sandboxes.home" property/placeholder.
cluster.node.sendinfo.interval	int	2000	time interval in ms; each node sends heart-beat with info about itself to another nodes; this interval specified how often the info is sent under common circumstances
cluster.node.sendinfo.cluster.node.sendinfo.min_interval	int	500	time interval in ms; Specified minimal interval between two heart-beats. Heart-beat may be send more often then specified by cluster.node.sendinfo.interval, e.g. when jobs start or finish. However the interval will never be shorter then this minimum.
cluster.node.sendinfo.history.interval	int	240000 (4 minutes)	time interval in ms, for which each node stores heart-beat in the memory; It's used for rendering figures in the web GUI-monitoring section
cluster.node.remove.interval	int	15000	time interval in ms; if no node info comes in this interval, node is considered as lost and it is removed from the cluster
cluster.max_allowed_time_shift_between_nodes	int	2000	Max allowed time shift between nodes. All nodes must have system time synchronized. Otherwise cluster may not work properly. So if this threshold is exceeded, node will be set as invalid.
cluster.group.name	String	cloverCluster	Each cluster has its unique group name. If you need 2 clusters in the same network environment, each of them would have its own group name.
cluster.jgroups.protocol.AUTH.value	String		Authentication string/password used for verification cluster nodes accessing the group. If this property is not specified,

property	type	default	description
			Cluster should be protected by firewall settings.
cluster.datasource.type	String	local	Change this property to "remote" if the node doesn't have direct connection to the CloverETL Server database, so it has to use some other cluster node as proxy to handle persistent operations. In such case, also property "cluster.datasource.delegate.nodeIds" must be properly configured. Properties jdbc.* will be ignored. Please note, that scheduler is active only on nodes with direct connection.
cluster.datasource.delegate.nodeIds	String		List of cluster node IDs (separated by comma ",") which this node may use as proxy to handle persistent operations. At least one of the listed node IDs must be running, otherwise this node will fail. All listed node IDs must have direct connection to CloverETL Server database properly configured. Property "cluster.datasource.delegate.nodeIds" is ignored by default. Property "cluster.datasource.type" must be set to "remote" to enable the feature.

Example of 2 node cluster configuration

This section contains examples of CloverETL cluster nodes configuration. We assume that the user "clover" is running the JVM process and the license will be uploaded manually in the web GUI. In addition it is necessssary to configure:

- sharing or replication of file system directory which the property "sandboxes.home" is pointing to. E.g. on unix-like systems it would be typically `/home/[username]/CloverETL/sandboxes`.
- connection to the same database from both nodes

Basic 2-nodes Cluster Configuration

This example describes the simple cluster: each node has direct connection to database.

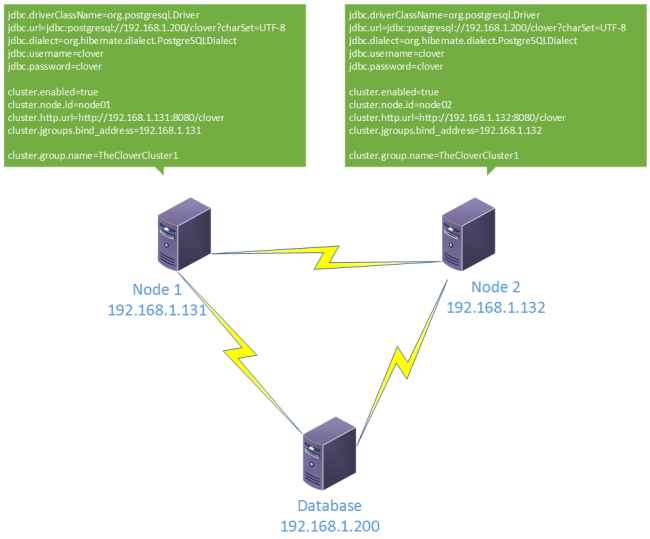


Figure 28.1. Configuration of 2-nodes cluster, each node has access to database

configuration of node on 192.168.1.131

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://192.168.1.200/clover?charSet=UTF-8
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.username=clover
jdbc.password=clover

cluster.enabled=true
cluster.node.id=node01
cluster.http.url=http://192.168.1.131:8080/clover
cluster.jgroups.bind_address=192.168.1.131

cluster.group.name=TheCloverCluster1
```

Configuration of node on 192.168.1.132

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://192.168.1.200/clover?charSet=UTF-8
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.username=clover
jdbc.password=clover

cluster.enabled=true
cluster.node.id=node02
cluster.http.url=http://192.168.1.132:8080/clover
cluster.jgroups.bind_address=192.168.1.132

cluster.group.name=TheCloverCluster1
```

If you use **Apache Tomcat**, the configuration is placed in `$CATALINA_HOME/webapps/clover/WEB-INF/config.properties` file. The location and file name on other application server may differ.

2-nodes Cluster with Proxied Access to Database

This cluster configuration is similar to previous one, but only one node has direct access to database. The node2 has to use node1 as a proxy.

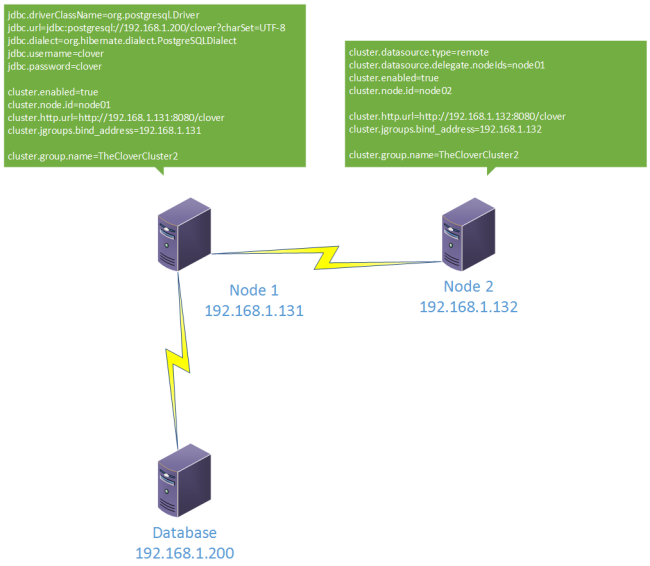


Figure 28.2. Configuration of 2-nodes cluster, one node without direct access to database

Configuration of node on 192.168.1.131

```
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://192.168.1.200/clover?charSet=UTF-8
jdbc.dialect=org.hibernate.dialect.PostgreSQLDialect
jdbc.username=clover
jdbc.password=clover

cluster.enabled=true
cluster.node.id=node01
cluster.http.url=http://192.168.1.131:8080/clover
cluster.jgroups.bind_address=192.168.1.131

cluster.group.name=TheCloverCluster2
```

Configuration of node on 192.168.1.132

```
cluster.datasource.type=remote                              (1)
cluster.datasource.delegate.nodeIds=node01

cluster.enabled=true
cluster.node.id=node02
cluster.http.url=http://192.168.1.132:8080/clover
cluster.jgroups.bind_address=192.168.1.132

cluster.group.name=TheCloverCluster2
```

1 These two lines describe acces to database via another node.

2-nodes cluster with load balancer

If you use any external load balancer, the configuration of CloverETL Cluster will be same as in the first example.

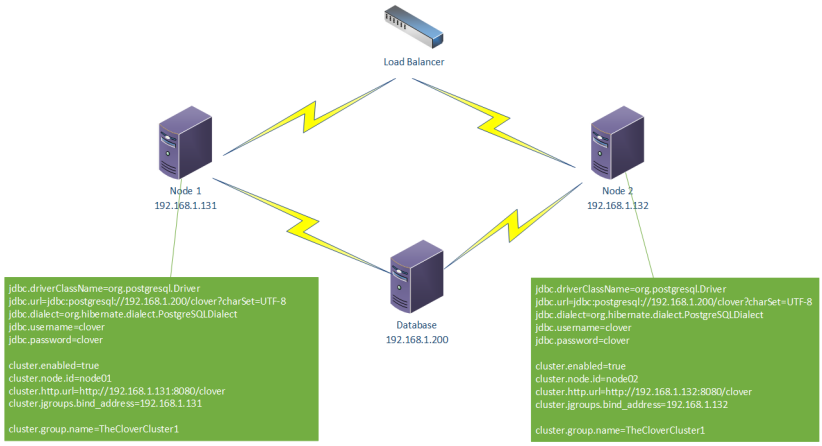


Figure 28.3. Configuration of 2-nodes cluster, one node without direct access to database

The `cluster.http.url` and `cluster.jgroups.bind_address` are urls of particular cluster nodes even if you use load balancer.

Jobs Load balancing properties

Multiplicators of load balancing criteria. Load balancer decides which cluster node executes graph. It means, that any node may process request for execution, but graph may be executed on the same or on different node according to current load of the nodes and according to these multiplicators.

The higher number, the higher relevance for decision. All multiplicators must be greater then 0.

Each node of the cluster may have different load balancing properties. Any node may process incomming requests for transformation execution and each may apply criteria for loadbalancing in a different way according to its own configuration.

These properties aren't vital for cluster configuration - default values are sufficient

Table 28.4. Load balancing properties

property	type	default	description
cluster.lb.balance.running_graphs	float	3	Specify importance of running graphs for load balancing.
cluster.lb.balance.memused	float	0.5	Specify importance of used memmory for load balancing.
cluster.lb.balance.cpus	float	1.5	Specify importance of number of CPUs for load balancing.
cluster.lb.balance.request_bonus	float	2	Specify importance of the fact, that the node is the same which processes request for execution. The same node, which decides where to execute graph. If you specify this multiplicator great enough, it will cause, that graph will be always executed on the same node, which processes request for execution.
cluster.lb.balance.node_bonus	float	1	Overall ratio bonus for configured node. Values greater then "1" increase probability the node will be choosen by the loadbalancer. Value "1" means no bonus or penalty. "0" means that the node will be never choosen by the loadbalancer, however it still may execute graphs, e.g. when there is no other node in cluster, or when the graph is designed to run on the node.

Running More Clusters

If you run more clusters, each cluster has to have its own unique name. If the name is not unique, the cluster nodes of different clusters may consider foreign cluster nodes as part of the same cluster. The cluster name is configured using `cluster.group.name` option. See Optional properties (p. 195).

Cluster reliability in unreliable network environment

CloverETL Server instances must cooperate with each other to form a cluster together. If the connection between nodes doesn't work at all, or if it's not configured, cluster can't work properly. This chapter describes cluster nodes behavior in environment, where the connection between nodes is somehow unreliable.

Nodes use three channels to exchange status info or data

1. synchronous calls (via HTTP/HTTPS)

   Typically nodeA requests some operation on nodeB, e.g. job execution. HTTP/HTTPS is also used for streaming data between workers of parallel execution

2. asynchronous messaging (TCP connection on port 7800 by default)

   Typically heart-beat or events, e.g. job started or finished.

3. shared database – each node must be able to create DB connection

   Shared configuration data, execution history etc.

Following scenarios are described below one by one, however they may occur together:

- nodeA can't establish HTTP connection to nodeB
- nodeA can't establish TCP connection (port 7800 by default) to nodeB
- nodeB is killed or it can't connect to the database
- long-term network malfunction may cause hang-on jobs

NodeA can't establish HTTP connection to nodeB

When HTTP request can't be established between nodes, jobs which are delegated between nodes, or jobs running in parallel on more nodes will fail. The error is visible in the executions history. Each node periodically executes check-task which checks HTTP connection to other nodes. If the problem is detected, one of the nodes is suspended, since they can't cooperated with each other.

Time-line describing the scenario:

- 0s network connection between nodeA and nodeB is down
- 0-40s a check-task running on nodeA can't establish HTTP connection to nodeB; check may last for 30s until it times-out; there is no re-try, if connection fails even just once, it's considered as unreliable, so the nodes can't cooperate
- status of nodeA or nodeB (the one with shorter uptime) is changed to “suspended”
- suspended node must be manually resumed, when the connection is recovered

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.check.checkMinInterval` - periodicity of cluster node checks (40000ms by default)
- `cluster.sync.connection.readTimeout` – HTTP connection timeout (30000ms by default)

NodeA can't establish TCP connection (port 7800 by default) to NodeB

TCP connection is used for asynchronous messaging. When the nodeB can't send/receive asynchronous messages, the other nodes aren't notified about started/finished jobs, so parent jobflow running on nodeA keep waiting for the event from nodeB. Also heart-beat is vital for meaningful load-balancing. The same check-task mentioned above also checks heart-beat from all cluster nodes.

Time-line describing the scenario:

- 0s network connection between nodeA and nodeB is down
- 60s nodeA uses the last available nodeB heart-beat
- 0-40s check-task running on nodeA detects missing heart-beat from nodeB
- status of nodeA or nodeB (the one with shorter uptime) is changed to “suspended”
- suspended node must be manually resumed, when the connection is recovered

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.check.checkMinInterval` - periodicity of cluster node checks (40000ms by default)
- `cluster.node.sendinfo.interval` – periodicity of heart-beat messages (2000ms by default)
- `cluster.node.sendinfo.min_interval` – the heart-beat may occasionally be sent more often than specified by “cluster.node.sendinfo.interval”, this property specifies minimum interval (500ms by default)
- `cluster.node.remove.interval` – maximum interval for missing heart-beat (60000ms by default)

NodeB is killed or it can't connect to the database

Access to the database is vital for running jobs, running scheduler and cooperation with other nodes also touching database is used for detection of dead process. When the JVM process of nodeB is killed, it stops touching the database and the other nodes may detect it.

Time-line describing the scenario:

- 0s-30s last touch on DB
- nodeB or its connection to the database is down
- 90s nodeA sees the last touch
- 0-40s check-task running on nodeA detects obsolete touch from nodeB
- status of nodeB is changed to “stopped”, jobs running on the nodeB are "solved", which means, that their status is changed to UNKNOWN and event is dispatched among the cluster nodes. Job result is considiered as error.

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.touch.interval` – periodicity of database touch (30000ms by default)
- `cluster.node.touch.forced_stop.interval` – interval when the other nodes accept last touch (90000ms by default)
- `cluster.node.check.checkMinInterval` - periodicity of cluster node checks (40000ms by default)
- `cluster.node.touch.forced_stop.solve_running_jobs.enabled` - not interval, but boolean value, which can switch the "solving" of running jobs mentioned above

Long-term network malfunction may cause hang-on jobs

Jobflow or master execution executing child jobs on another cluster nodes must be notified about status changes of their child jobs. When the asynchronous messaging doesn't work, events from the child jobs aren't delivered, so parent jobs keep running. When the network works again, the child job events may be re-transmitted, so hung parent job may be finished. However the network malfunction may be so long, that the event can't be re-transmitted.

Please see following time-line to consider proper configuration:

- job A running on nodeA executes job B running on nodeB
- network between nodeA and nodeB is down from some reason
- job B finishes and sends the “finished” event, however it can't be delivered to nodeA – event is stored in the “sent events buffer”
- Since the network is down, also heart-beat can't be delivered and maybe HTTP connections can't be established, the cluster reacts as described in the sections above. Even though the nodes may be suspended, parent job A keeps waiting for the event from job B

now, there are 3 possibilities:

a. Network finally starts working and since all undelivered events are in the “sent events buffer”, they are re-transmitted and all of them are finally delivered. Parent job A is notified and proceeds. It may fail later, since some cluster nodes may be suspended.

b. Network finally starts working, but number of the events sent during the malfunction exceeded “sent events buffer” limit size. So some messages are lost and won't be re-transmitted. Thus the buffer size limit should be higher in the environment with unreliable network. Default buffer size limit is 10000 events. It should be enough for thousands of simple job executions, basically it depends on number of job phases. Each job execution produces at least 3 events (job started, phase finished, job finished). Please note that there are also some other events fired occasionally (configuration changes, suspending, resuming, cache invalidation). Also messaging layer itself stores own messages to the buffer, but it's just tens messages in a hour. Heart-beat is not stored in the buffer.

   There is also inbound events buffer used as temporary storage for events, so the events may be delivered in correct order when some events can't be delivered at the moment. When the cluster node is inaccessible, the inbound buffer is released after timeout, which is set to 1 hour by default.

c. Node B is restarted, so all undelivered events in the buffer are lost.

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.jgroups.protocol.NAKACK.gc_lag` – limit size of the sent events buffer; Please note that each stored message takes 2kB of heap memory (default limit is 10000 events)
- `cluster.jgroups.protocol.NAKACK.xmit_table_obsolete_member_timeout` – inbound buffer timeout of unaccessible cluster node

Chapter 29. Recommendations for Cluster Deployment

1. All nodes in the cluster should have a synchronized system date-time.
2. All nodes share sandboxes stored on a shared or replicated filesystem. The filesystem shared among all nodes is single point of failure. Thus, the use of a replicated filesystem is strongly recommended.
3. All nodes share a DB, thus it must support transactions. I.e. The MySQL table engine, MyISAM, may cause strange behaviour because it is not transactional.
4. All nodes share a DB, which is a single point of failure. Use of a clustered DB is strongly recommended.
5. Configure the license by "license.file" property or upload it in the Web GUI, so it's stored in the database. Do not use `clover-license.war`.

List of Figures

List of Tables

List of Examples