CloverETL Server

Reference Manual

CloverETL

CloverETL Server: Reference Manual

This Reference Manual refers to CloverETL Server 4.6.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 e-mail to `support@cloveretl.com`.

Consider How to speed up communication with CloverCARE support before contacting the support.

Table of Contents

I. CloverETL Server 1
  1. What is CloverETL Server? 2
II. Installation Instructions 4
  2. System Requirements for CloverETL Server 5
  3. Installing 7
    Evaluation Server 9
      Installation 9
    Production Server 12
      Apache Tomcat 13
      Jetty 19
      IBM WebSphere 23
      GlassFish / Sun Java System Application Server 27
      JBoss Application Server 30
      JBoss Enterprise Application Platform 35
      Oracle WebLogic Server 41
    CloverETL Server Activation 45
      CloverETL Server Activation Alternatives 47
    IBM InfoSphere MDM Plugin Installation 48
      Installation into Server 48
    Possible Issues during Installation 50
  4. Postinstallation Configuration 55
    Memory Settings 55
    Maximum Number of Open Files 57
    Firewall Exceptions 57
  5. Upgrading Server to Newer Version 58
III. Configuration 60
  6. Configuration Sources and Their Priorities 61
    Configuration Sources 61
      Environment Variables 61
      System Properties 61
      Properties File on Default Location 61
      Properties File on Specified Location 61
      Modification of Context Parameters in web.xml 61
      Context Parameters (Available on Apache Tomcat) 62
    Priorities of Configuration Sources 62
  7. Setup 63
    Using Setup 63
      Setup Tabs 65
  8. Examples of DB Connection Configuration 71
    Embedded Apache Derby 72
    MySQL 73
      Creating database 73
      CloverETL setup 73
    DB2 74
      Creating database 74
      DB2 on Linux/Windows 74
      DB2 on AS/400 76
    Oracle 77
      Creating database 77
      CloverETL setup 77
    Microsoft SQL Server 78
      Creating database 78
      CloverETL setup 78
    Postgre SQL 80
      Creating database 80

CloverETL setup 80
JNDI DB DataSource 81
Encrypted JNDI 82
9. List of Properties 88
10. Secure Configuration Properties 94
11. Logging 98
IV. Administration 100
12. Temp Space Management 101
Overview 101
Management 102
13. Secure Parameters 107
14. Users and Groups 110
LDAP Authentication 111
Users 114
Groups 117
15. Server Side Job Files - Sandboxes 129
Referencing Files from the ETL Graph or Jobflow 131
Sandbox Content Security and Permissions 132
Sandbox Content 133
Job Config Properties 137
WebDAV Access to Sandboxes 140
WebDAV Clients 140
WebDAV Authentication/Authorization 140
16. CloverETL Server Monitoring 142
Standalone Server Detail 142
Cluster Overview 147
Node Detail 148
Server Logs 149
17. Server Configuration Migration 150
Server Configuration Export 151
Server Configuration Import 152
18. Diagnostics 156
V. Using Graphs 157
19. Graph/Jobflow Parameters 158
Parameters by Execution Type 159
Executed from Web GUI 159
Executed by Launch Service Invocation 159
Executed by HTTP API Run Graph Operation Invocation 159
Executed by RunGraph Component 159
Executed by WS API Method executeGraph Invocation 159
Executed by Task "graph execution" by Scheduler 159
Executed from JMS Listener 159
Executed by Task "Start a graph" by Graph/Jobflow Event Listener 159
Executed by Task "graph execution" by File Event Listener 160
Adding Another Graph Parameters 161
Additional "Graph Config Parameters" 161
Task "execute_graph" Parameters 161
20. Tasks 162
Send an Email 163
Placeholders 164
Execute Shell Command 167
Start a Graph 170
Start a Jobflow 172
Start a Profiler Job 173
Abort job 174
Archive Records 175
Send a JMS Message 177
Execute Groovy Code 180

21. Manual Task Execution ... 182
22. Scheduling ... 183
  Timetable Setting ... 184
  Allocations of Scheduled Task on Nodes ... 189
  Scheduling the Tasks - Examples ... 190
23. Viewing Job Runs - Executions History ... 191
  Filtering and ordering ... 191
24. Listeners ... 194
  Graph Event Listeners ... 195
    Graph Events ... 195
    Listener ... 196
    Tasks ... 197
    Use Cases ... 197
  Jobflow Event Listeners ... 202
    Jobflow Events ... 202
    Listener ... 203
    Tasks ... 203
  JMS Messages Listeners ... 205
  Universal Event Listeners ... 210
    Evaluation Criteria ... 210
  File Event Listeners (remote and local) ... 212
    Cluster environment ... 212
    Supported filesystems and protocols ... 213
    Observed file ... 214
    File Events ... 215
    Check Interval, Task and Use Cases ... 215
    Howtos ... 216
  Task Failure Listeners ... 218
    Task Choice ... 218
    Task Failed E-mail Template ... 218
25. API ... 220
  Simple HTTP API ... 220
  JMX mBean ... 230
    JMX Configuration ... 230
    Operations ... 232
  SOAP WebService API ... 233
    SOAP WS Client ... 233
    SOAP WS API Authentication/Authorization ... 233
  Launch Services ... 234
    Launch Services Authentication ... 239
    Sending the Data to Launch Service ... 239
    Results of the Graph Execution ... 239
  CloverETL Server API Extensibility ... 241
    Groovy Code API ... 241
26. Recommendations for Transformations Developers ... 243
27. Extensibility - CloverETL Engine Plugins ... 244
28. Troubleshooting ... 245
VI. Cluster ... 246
29. Clustering Features ... 247
  High Availability ... 247
  Scalability ... 247
    Transformation Requests ... 248
    Parallel Data Processing ... 248
  Graph Allocation Examples ... 254
  Example of Distributed Execution ... 255
    Details of the Example Transformation Design ... 255
    Scalability of the Example Transformation ... 258
30. Cluster Configuration ... 260

Mandatory Properties 261
Optional Properties 263
Example of 2 Node Cluster Configuration 266
Basic 2-nodes Cluster Configuration 266
2-nodes Cluster with Proxied Access to Database 267
2-nodes cluster with load balancer 268
Jobs Load Balancing Properties 269
Running More Clusters 270
Cluster Reliability in Unreliable Network Environment 271
NodeA Cannot Establish HTTP Connection to NodeB 271
NodeA Cannot Establish TCP Connection (Port 7800 by Default) to NodeB 271
NodeB is Killed or It Cannot Connect to the Database 272
Auto-Resuming in Unreliable Network 272
Long-Term Network Malfunction May Cause Jobs to Hang on 273
31. Recommendations for Cluster Deployment 274
32. Multiple CloverServer Instances on the same Host 275

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. 220).)	by executing external process or by calling Java API
engine initialization	during server startup	init is called for each graph execution
thread and memory optimization	threads recycling, graphs cache, etc.	not implemented
scheduling	scheduling by timetable, onetime trigger, logging included	external tools (i.e. Cron) can be 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 an email, execute a shell command or execute another graph. See details in Graph Event Listeners (p. 195) 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. 129).	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. 220).	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 horizontal scalability of transformation requests as well as data scalability. See details in Chapter 29, Clustering Features (p. 247) In addition CloverEngine implements is vertical scalability natively.	Clover Engine implements vertical 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

	Basic Edition	Corporate Edition	Cluster
RAM	4 GB (recommended 16 GB)	8 GB (recommended 64 GB)	8 GB (recommended 64 GB)
Processors	up to 4 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] Minimum value, the disk space depends on data.
[c] Disk space for shared sandboxes is required only for CloverETL Cluster.

Software Requirements

Operating system

CloverETL server is compatible with Windows and Unix-based systems, as well as with other systems supporting Java (Mac OS X, IBM System, etc.).

Java Virtual Machine

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

Application Server

- Apache Tomcat 6 or 7 or 8 (p. 13)
- Jetty 9.1 (p. 19)
- IBM WebSphere 8.5 (p. 23)
- Glassfish 3.1 (p. 27)
- JBoss 6 or 7 (p. 30)
- Oracle WebLogic 11g (10.3.6) or 12c (12.1.2 or 12.1.3) 32/64 bit (p. 41)

Table 2.2. CloverETL Server Compatibility Matrix

	CloverETL 3.5	CloverETL 4.0	CloverETL 4.1, 4.2, 4.3, 4.4, and 4.5		CloverETL 4.6	
Application Server	**Java 6 and 7**	**Java 7**	**Java 7**	**Java 8**	**Java 7**	**Java 8**
Tomcat 6[a]	✓	✓	✓	✓	✓	✓
Tomcat 7	✗	✓	✓	✓	✓	✓
Tomcat 8	✗	✗	✓	✓	✓	✓
Pivotal tc Server Standard (3.1.3, Tomcat 7)	✗	✗	✗	✓	✗	✓
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	✗	✓	✓[b]	✓[c]	✓[b]	✓[c]
Glassfish 2	✓	✗	✗	✗	✗	✗
Glassfish 3	✗	✓	✓	✗	✓	✗
WebSphere 7	✓	✗	✗	✗	✗	✗
WebSphere 8.5	✗	✓	✓	✗	✓	✗

[a] Please note that support for Apache Tomcat 6.0.x has ended on 31 December 2016. See End of life for Apache Tomcat 6.0.x for more information.
[b]EAP 6.2
[c]EAP 6.4

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

Database servers

We support the following database servers. The officially supported versions, we are testing against, are in parentheses.

- MySQL (5.6.12) (p. 73)
- DB2 (10.5.1) (p. 74)
- Oracle (11.2.0.2.0) (p. 77)
- SQL Server 2008 (10.0.1600.22) (p. 78)
- PostgreSQL (9.2.4) (p. 80)

Chapter 3. Installing

This chapter describes two different server installations - Evaluation Server (p. 9) and Production Server (p. 12) - and provide instructions on installing the CloverETL Server License.

Evaluation Server

The Evaluation Server (p. 9) consists of CloverETL Server bundled with the Tomcat application container. The server performs basic configuration during the first startup and requires no additional database server. This option is **recommended only for basic evaluation** of CloverETL Server's functions.

However with further configuration, it is possible to make the evaluation server **ready for production environment**. This process requires connection to an external, dedicated database and subsequent configuration of services (e.g. SMTP, LDAP, etc.).

Important

The Apache Derby DB, bundled with the evaluation server, is **not** recommended for production environment. Please use one of the supported external databases.

Production Server

In case of Production Server (p. 12) the CloverETL Server is installed on one of the several compatible application containers. This process requires additional configuration (e.g. memory allocation, database connection, etc.) but allows you to choose an application container and external database according to your preference.

Installation and configuration procedure

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

Install an application server

CloverETL Server is compatible with several application containers. Following subsections offer detailed instructions on installation of the respective application servers and their subsequent configuration.

Set up limits on a number of opened files and memory allocation

CloverETL Server's graph transformations and evaluations may require more memory than the default limit set in the database as well as higher number of simultaneously opened files. These instructions provide recommendation on adjusting both the Memory Settings (p. 55) and the Maximum Number of Open Files (p. 57).

Install CloverETL Server into application server

CloverETL Server is provided as a web archive (`.war`) file for an easy deployment.

Create a database dedicated to CloverETL server

Unlike the Evaluation server, the Production server requires that you have created a dedicated database for CloverETL Server. In the configuration phase of this manual, you will be guided to Chapter 8, Examples of DB Connection Configuration (p. 71) with instructions on how to properly configure the properties file of various databases.

Set up connection to the database

The CloverETL Server Console GUI lets you configure a number of items including database connection, license file, etc. Optionally, you can set up password encryption in configuration files for higher security. For details, see Chapter 7, Setup (p. 63).

Install a license

To be able to execute graphs, you need to install a valid license. There are three options for CloverETL Server Activation (p. 45).

Perform additional server configuration

Set up a master password for secure parameters

When handling sensitive information (e.g. passwords), it is advised to define secure graph parameters. This action requires a master password (see Chapter 13, Secure Parameters (p. 107)).

Set up SMTP server connection

CloverETL Server lets you configure an SMTP connection for reporting events on the server via e-mails.

Configure temp space

CloverETL Server works with temporary directories and files. To ensure the components work correctly, you should configure the `Temp space` location on the file system. For details, see Chapter 12, Temp Space Management (p. 101).

Configure sandboxes

Lastly, you should set the content security and user's permissions for sandboxes. For details and instructions, see Chapter 15, Server Side Job Files - Sandboxes (p. 129).

Evaluation Server

The default installation of **CloverETL Server** uses the embedded Apache Derby DB; therefore, it does not require any extra database server. Furthermore, it does **not require** any subsequent **configuration**, as CloverETL Server configures itself during the first startup. Database tables and some necessary records are automatically created on the first startup with an empty database.

By performing a subsequent configuration, you can evaluate other CloverETL Server features (e.g. sending e-mails, LDAP authentication, clustering, etc.). This way, you can also prepare the evaluation server for production environment. However, note that the embedded Apache Derby database is **not** recommended for production environment. Therefore, before the subsequent configuration, choose one of the supported external dedicated database.

If the CloverETL Server must be evaluated on application containers other than Tomcat, or you prefer a different database, proceed with the common installation of Production Server (p. 12)

Note

Default login credentials for CloverETL Server Console are:

Username: **clover**

Password: **clover**

Installation

1. Make sure you have a compatible Java version:

   Important

   CloverETL Server 4.1 and higher requires Oracle JDK or JRE v. **1.7.x** or **higher**. We recommend JDK 1.8.x.

   - You can check your installed Java version by typing the following command to the command prompt or terminal:

```
java -version
```

   - Alternatively, for **macOS** and **Windows** platforms, see How to find Java version in Windows or Mac.

2. Download and extract the CloverETL Evaluation Server.

   - Go to CloverETL User Login Page.
   - Using your credentials, log into your account, navigate to the download section and download the CloverETL Evaluation Server Bundle.
   - Extract the `.zip` archive. (For example, the name of an archive containing `CloverETL Server v 4.5.0` bundled with `Tomcat v 8.0.30` will be `CloverETLServer.4.5.0.Tomcat-8.0.30.zip`.)

   Note

   It is recommended to place the extracted content on a path that does not contain space character(s).

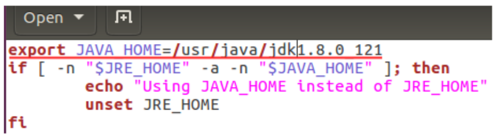
   `C:\Program Files` or `/home/user/some dir` ✖

`C:\Users\Username` or `/home/user/some_dir`

3. Set the `JAVA_HOME` or `JRE_HOME` Environment Variables.

   - **Unix-like systems:**
     - Using a text editor, open the `setenv.sh` file located in the `[Tomcat_home]/bin/` directory.
     - Define the path at the beginning of the file (the path may differ):

```
export JAVA_HOME=/usr/jdk1.8.0_121
```

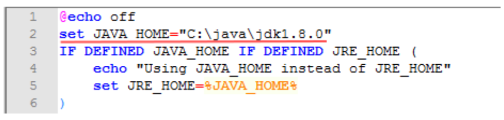


```
export JAVA_HOME=/usr/java/jdk1.8.0_121
if [ -n "$JRE_HOME" -a -n "$JAVA_HOME" ]; then
        echo "Using JAVA_HOME instead of JRE_HOME"
        unset JRE_HOME
fi
```

Figure 3.1. `setenv.sh` edited in Linux.

   - **Windows system:**
     - Using a text editor, open the `setenv.bat` file located in the `[Tomcat_home]\bin\` directory.
     - Define the path on the second line of the file (the path may differ):

```
set JAVA_HOME="C:\java\jdk1.8.0"
```



```
@echo off
set JAVA_HOME="C:\java\jdk1.8.0"
IF DEFINED JAVA_HOME IF DEFINED JRE_HOME (
    echo "Using JAVA_HOME instead of JRE_HOME"
    set JRE_HOME=%JAVA_HOME%
)
```

Figure 3.2. `setenv.bat` edited in Windows.

4. Run Tomcat.

   - **Unix-like systems:**

     Run the `[Tomcat_home]/bin/startup.sh` file.

   - **Windows system:**

     Run the `[Tomcat_home]\bin\startup.bat` file.

5. Check whether CloverETL Server is running.

   - Open a new tab in your browser and type http://localhost:8083/clover/ in the address bar.
   - Use the **default administrator credentials** to access the web GUI: username: **clover**, password: **clover**.

Note

If you access the web GUI of the CloverETL Server before the product activation, you will be asked to install the CloverETL Server license key.

Continue with: CloverETL Server Activation (p. 45)

Tip

To safely stop the server, run the `[Tomcat_home]/bin/shutdown.sh` or `[Tomcat_home]\bin\shutdown.bat` file for Unix-like or Windows system respectively.

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

Production Server

This section describes in detail the installation of CloverETL Server on various application containers and its subsequent configuration required for production environment. For simple evaluation of CloverETL Server features use Evaluation server (p. 9) (note that CloverETL Evaluation server can also be configured for production use).

CloverETL Server for production environment is shipped as a *Web application archive* (WAR file), and uses an external, dedicated database. 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 following chapters.

List of Suitable Containers

- Apache Tomcat (p. 13)
- Jetty (p. 19)
- IBM WebSphere (p. 23)
- GlassFish / Sun Java System Application Server (p. 27)
- JBoss Application Server (p. 30)
- Oracle WebLogic Server (p. 41)

In case of problems during the installation see Possible Issues during Installation (p. 50).

Important

CloverETL Server 4.1 and higher requires Oracle JDK or JRE v. **1.7.x** or **higher**. We recommend JDK 1.8.x.

Apache Tomcat

Installation of Apache Tomcat (p. 13)
Apache Tomcat as a Windows Service (p. 14)
Apache Tomcat on IBM AS/400 (iSeries) (p. 16)
Installation of CloverETL Server (p. 16)
Configuration of CloverETL Server on Apache Tomcat (p. 17)

Important

See Application Server (p. 5) in system requirements for currently supported **Apache Tomcat** versions and required **Java** versions.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Please note that since 31 December, 2016, the Apache Tomcat team **has ended Apache Tomcat 6.0.x support**. See End of life for Apache Tomcat 6.0.x for more information.

Installation of Apache Tomcat

1. Download the binary distribution: Tomcat 6[1], Tomcat 7 or Tomcat 8.

   CloverETL Server is developed and tested with the Apache Tomcat 6.0.x, 7.0.x and 8.0.x containers. Running the Server with other versions may result in unpredictable behavior.

2. Extract the downloaded archive (`zip` or `tar.gz`).

3. Set up `JAVA_HOME` to point to the correct Java version.

4. Run Tomcat.

   - **Unix-like systems:**

     Run the `[Tomcat_home]/bin/startup.sh` file.

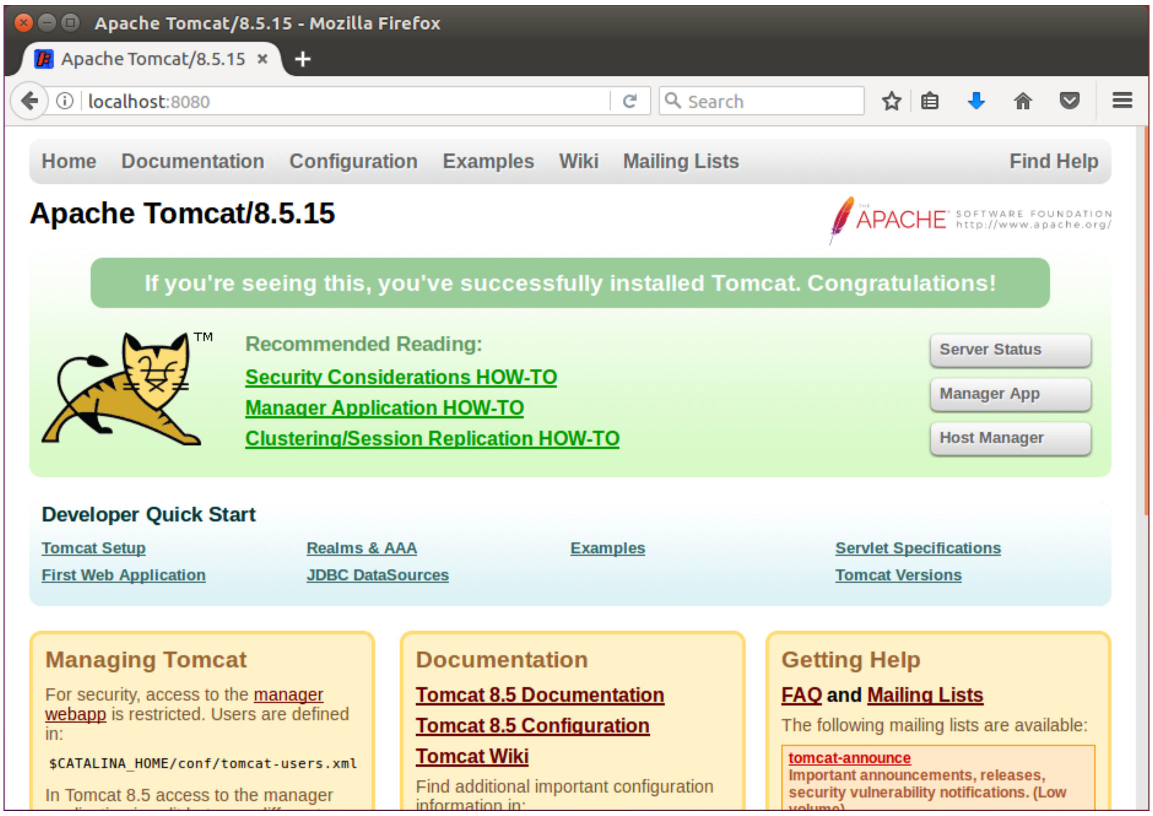
   - **Windows system:**

     Run the `[Tomcat_home]\bin\startup.bat` file.

5. Check whether Tomcat is running.

   - Open a new tab in your browser and type http://localhost:8080/ in the address bar.

     If the Apache Tomcat Information page appears (see below), the server is successfully installed:



Apache Tomcat/8.5.15

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Developer Quick Start
Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Managing Tomcat
For security, access to the manager webapp is restricted. Users are defined in:
`$CATALINA_HOME/conf/tomcat-users.xml`
In Tomcat 8.5 access to the manager

Documentation
Tomcat 8.5 Documentation
Tomcat 8.5 Configuration
Tomcat Wiki
Find additional important configuration

Getting Help
FAQ and Mailing Lists
The following mailing lists are available:
tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low

Figure 3.3. Apache Tomcat welcome page

Tip

For detailed installation instructions, see: Tomcat 6, Tomcat 7 or Tomcat 8 Setup Guide.

Note

For the installation on **IBM AS/400**, continue with Apache Tomcat on IBM AS/400 (iSeries) (p. 16).

Continue with: Installation of CloverETL Server (p. 16).

Apache Tomcat as a Windows Service

1. Download the **32-bit/64-bit Windows Service Installer** file in the **Binary Distributions** section on the Tomcat 6[1], Tomcat 7 or Tomcat 8 download page.

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

3. Check whether Tomcat is running.

   - Type http://localhost:8080/ in your browser's address bar.

   - If the Apache Tomcat Information page appears, the server is successfully installed.

4. When Tomcat is installed as a Windows service, CloverETL is configured by one of the following options:

Graphical configuration utility

- Run the `[Tomcat_home]\bin\Tomcat8w.exe` file.

- In the **Apache Tomcat Properties** dialog box, select the **Java** tab and set the initial and maximum heap size in **Initial memory pool** and **Maximum memory pool** fields to 512MB and 1024MB respectively. Other configuration parameters can be defined in **Java Options** field, being separated by new line.
- Click on **Apply** and restart the service.

Note

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

Command Prompt tool

- Run the `[Tomcat_home]\bin\Tomcat8.exe` file.
- If Tomcat is running, navigate to `[Tomcat_home]\bin` and stop the service by typing:

```
.\Tomcat8.exe //SS//Tomcat8
```

in the Command Prompt. (When using different version of Tomcat, change the number in the command to reflect the installed version.)

- Configure the service by typing the command:

```
.\Tomcat8.exe //US//Tomcat8 --JvmMs=512 --JvmMx=1024 --JvmOptions=-Dclover.config.file=C:\path\to\clover-config.pr
```

The parameter JvmMs is the initial and JvmMx is the maximum heap size in MB; `JvmOptions` are separated by '#' or ';'.

Important

If you use Java 7, change `-XX:MaxMetaspaceSize` to `-XX:MaxPermSize`.

- Start the service from Windows administration console or by typing the following command in the Command Prompt:

```
.\Tomcat8.exe //TS//Tomcat8
```

Tip

By default, when Apache Tomcat is run as a Windows service, it is **not available** for Java process monitoring tools (e.g., **JConsole** or **JVisualVM**). However, these 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 these options, you can connect to **port 3333** using JMX and monitor the server.

Note

The instructions can be applied to Tomcat 6 and Tomcat 7, simply replace the number 8 in the file name with the number 6 or 7.

More information about running Java applications as Windows Service can be found at Apache Commons.

Continue with: Installation of CloverETL Server (p. 16).

Apache Tomcat on IBM AS/400 (iSeries)

Additional settings are required to run CloverETL Server on the iSeries platform:

1. Declare you are using Java 7.0 32-bit.

2. Run Java with parameter `-Djava.awt.headless=true`.

To configure the settings, modify (or create) the `[Tomcat_home]/bin/setenv.sh` file to contain:

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

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

Continue with: Installation of CloverETL Server (p. 16)

Installation of CloverETL Server

1. Check if you meet the prerequisites:

   - Oracle JDK or JRE is installed (See Java Virtual Machine (p. 5) for the required Java version.)
   - `JAVA_HOME` or `JRE_HOME` environment variable is set.
   - A supported version (p. 6) of Apache Tomcat[1] is installed.

2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters:

   Unix-like systems:

   - Create the `[Tomcat_home]/bin/setenv.sh` file.
   - Type or paste in the following lines:

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

   Windows systems:

   - Create the `[Tomcat_home]\bin\setenv.bat` file.

- Type or paste in the following lines:

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

Important

If you use Java 7, change `-XX:MaxMetaspaceSize` to `-XX:MaxPermSize`.

Tip

For performance reasons, it is recommended to run the container in the "server" mode by setting the `-server` switch, as seen in the settings above.

3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for Apache Tomcat.

4. Copy `clover.war` to the `[Tomcat_home]/webapps` directory.

Note

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

5. Tomcat should automatically detect and deploy the `clover.war` file.

6. Check whether CloverETL Server is running:

   - Run Tomcat.
   - Open a new tab in your browser and type http://localhost:8080/clover/ in the address bar.
   - Use the **default administrator credentials** to access the web GUI: username: **clover**, password: **clover**.

Continue with: Configuration of CloverETL Server on Apache Tomcat (p. 17)

Configuration of CloverETL Server on Apache Tomcat

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61,) or specified (p. 61) location.

Content of such a file (example with MySQL database):

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

Properties File in Specified Location

The properties file is loaded from a location specified by a system property or by an environment variable `clover_config_file` or `clover.config.file`.

1. Create the `cloverServer.properties` file in a directory readable by Apache Tomcat. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)

2. Edit the `[Tomcat_home]/bin/setenv.sh` file (if it does not exist, you may create it).

3. Set the system property by adding the following line into the file:

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

Note

Continue with: Chapter 4, Postinstallation Configuration (p. 55)

Jetty

Installation of Jetty (p. 19)
Installation of CloverETL Server (p. 20)
Configuration of CloverETL Server on Jetty (p. 21)

Important

See Application Server (p. 5) in system requirements for currently supported **Jetty** versions and required **Java** versions.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of Jetty

1. Download the Jetty release from the official download page.

   CloverETL Server is developed and tested with the Jetty 6.x.x and 9.x.x containers. Running the Server with other versions may result in unpredictable behavior.

2. Extract the downloaded archive (`zip` or `tar.gz`).

3. Run Jetty.

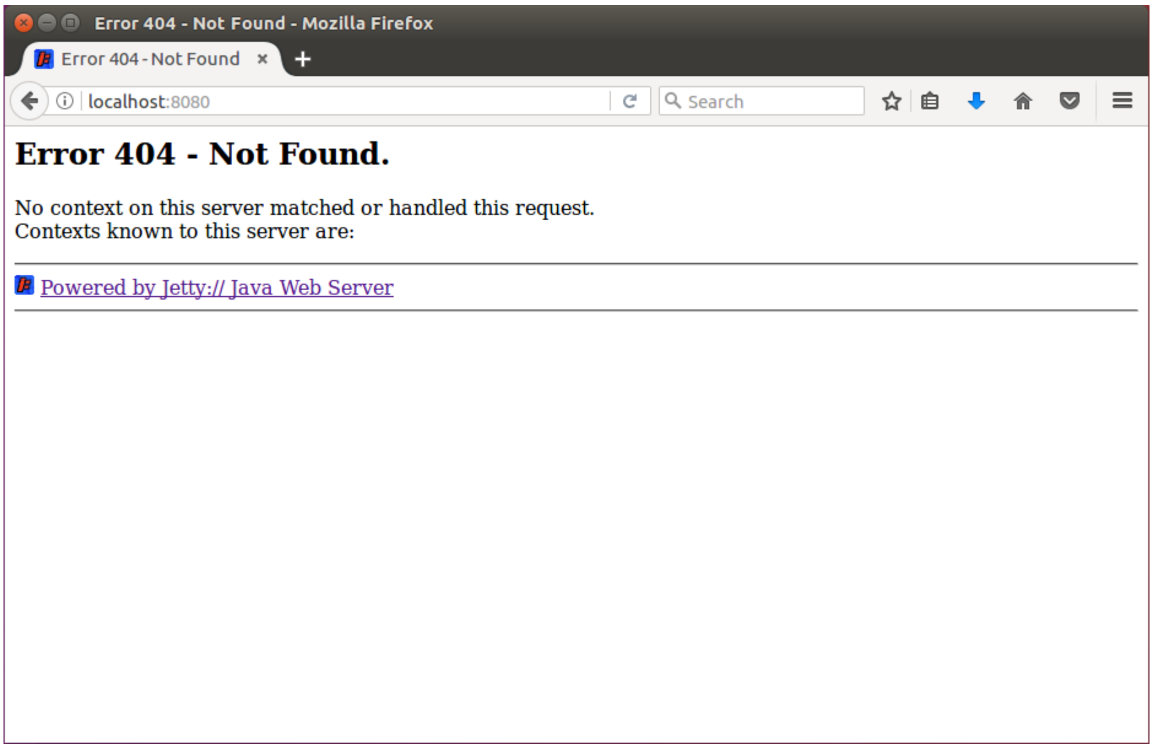
   - **Unix-like systems:**
     - Run `[Jetty_home]/bin/jetty.sh start`
   - **Windows system:**
     - Run the `[Jetty_home]\java -jar start.jar --exec` command in Windows command prompt.

4. Check whether Jetty is running.

   - Open a new tab in your browser and type http://localhost:8080/ in the address bar.

   Note

   Since the `clover.war` file is not yet implemented, you may see the **Error 404 - Not found** status code (see the figure below). However, it means that the server is running.



Error 404 - Not Found.

No context on this server matched or handled this request.
Contexts known to this server are:

Powered by Jetty:// Java Web Server

Figure 3.4. Jetty welcome page

Installation of CloverETL Server

1. Check if you meet the prerequisites:

   - Oracle JDK or JRE is installed (see Java Virtual Machine (p. 5) for the required Java version).
   - **JAVA_HOME** or **JRE_HOME** environmental variable is set.
   - A supported version (p. 6) of Jetty is installed.

2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters and **classloaders memory limit** by adjusting the "XX:MaxMetaspaceSize" parameter:

   - **Unix-like systems:**

     Edit the `[Jetty_home]/bin/jetty.sh` file.

     Type or paste the following line at the end of the file:

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

   - **Windows system:**

     Edit the `[Jetty_home]\start.ini` file.

     Type or paste the following line at the end of the file:

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

Important

If you use Java 7, change `-XX:MaxMetaspaceSize` to `-XX:MaxPermSize`.

3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for Jetty.

4. Copy `clover.war` to the `[Jetty_home]/webapps` directory.

5. Run Jetty.

   - **Unix-like systems:**
     - Run `[Jetty_home]/bin/jetty.sh start`
   - **Windows system:**
     - Run the `[Jetty_home]\java -jar start.jar --exec` command in Windows command prompt.

6. Check whether CloverETL Server is running:

   - Open a new tab in your browser and type http://localhost:8080/clover/ in the address bar.
   - Use the **default administrator credentials** to access the web GUI: username: **clover**, password: **clover**.

Configuration of CloverETL Server on Jetty

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61) or specified (p. 61) location.

Content of such a file (example with MySQL database):

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

Note

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

Properties file in Specified Location

The 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.

1. Create the `cloverServer.properties` file in a directory readable by Jetty. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)

2. **Unix-like systems:**

   a. Edit the `[Jetty_home]/bin/jetty.sh` file.

   b. Set the system property by adding the following line into the file:

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

   Windows system:

   a. Edit the `[Jetty_home]\start.ini` file.

   b. Set the system property by adding the following line into the file just after the memory settings:

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

Note

Continue with: Chapter 4, Postinstallation Configuration (p. 55)

IBM WebSphere

Installation of IBM Websphere (p. 23)
Installation of CloverETL Server (p. 23)
Configuration of CloverETL Server on IBM WebSphere (p. 25)

Important

See Application Server (p. 5) in system requirements for currently supported **IBM WebSphere** versions and required **Java** versions.

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 SDKs may lead to deadlocks during execution of specific ETL graphs.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of IBM Websphere

1. Create a My IBM account on https://www.ibm.com

2. Go to IBM Marketplace and download IBM Installation Manager.

3. Follow the instructions to download IBM Websphere.

   CloverETL Server is developed and tested with the IBM WebSphere 7 and 8.5 container. Running the Server with other versions may result in unpredictable behavior.

4. Create a profile.

   - **Unix-like systems:**

     Run the `[IBM_home]/WebSphere/AppServer/bin/ProfileManagement/pmt.sh`.

   - **Windows system:**

     Run the `[IBM_home]\WebSphere\AppServer\bin\ProfileManagement\pmt.bat`.

   Important

   Make sure the profile name does not contain the keyword "clover", otherwise the CloverETL server won't start properly.

5. Check whether the server is operational.

   You will be prompted to run a first-time server management tool that will check whether the installation was successful and the server can be started.

   Go to http://localhost:9060/ibm/console/ where you should be able to see the WebSphere login page.

Installation of CloverETL Server

1. Check if you meet the prerequisites:

   - IBM Java SDK is installed (see Java Virtual Machine (p. 5) for the required Java version).

   - **JAVA_HOME** or **JRE_HOME** environmental variable is set.

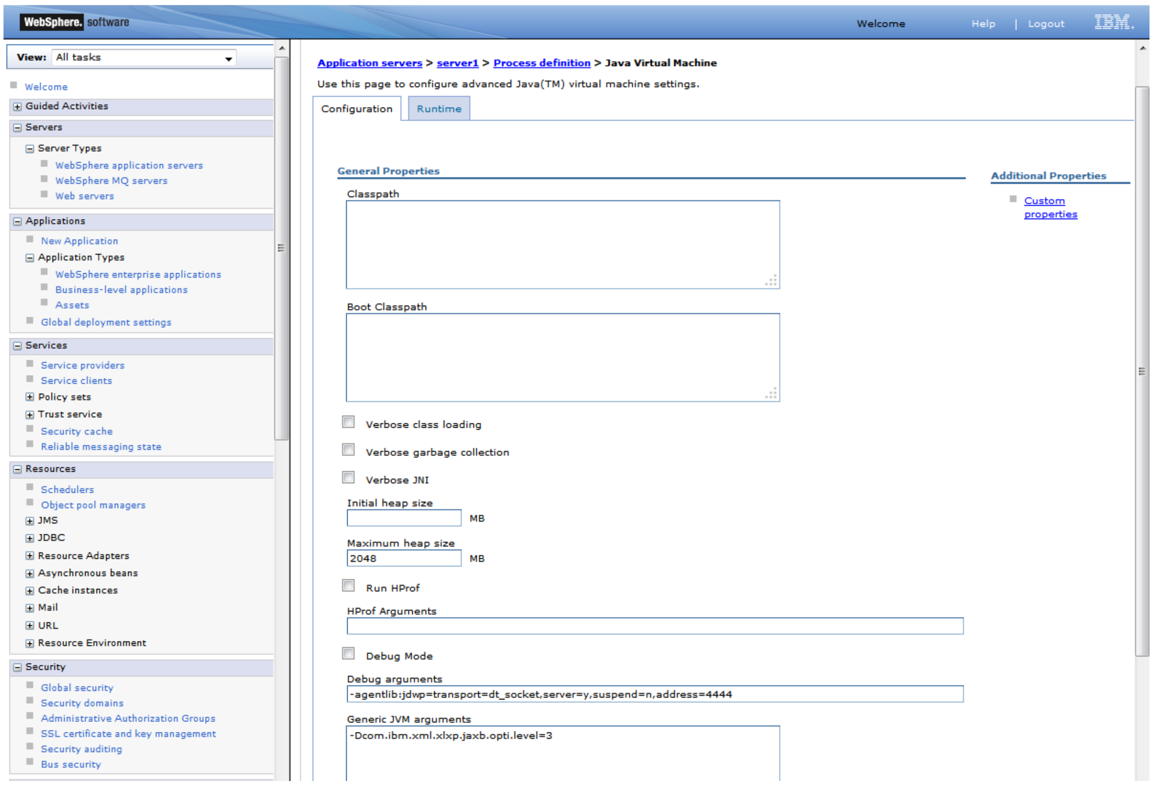
- A supported version (p. 6) of IBM WebSphere is installed.

2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the [Memory Settings](#) (p. 55) section).

   You can set the limits in IBM WebSphere's **Integrated Solutions Console** (default URL: http://localhost:9060/ibm/console/).

   - Go to **Servers →Server Types →WebSphere application servers →[Server_Name] (default name: server1) →Java and Process Management →Process definition →Java Virtual Machine**

   - Change the value in the **Maximum heap size** field to 2048 MB. The default value (256 MB) is insufficient for ETL transformations.



Application servers > server1 > Process definition > Java Virtual Machine

Use this page to configure advanced Java(TM) virtual machine settings.

General Properties

Classpath

Boot Classpath

Verbose class loading

Verbose garbage collection

Verbose JNI

Initial heap size MB

Maximum heap size 2048 MB

Run HProf

HProf Arguments

Debug Mode

Debug arguments -agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=4444

Generic JVM arguments -Dcom.ibm.xml.xlxp.jaxb.opti.level=3

Figure 3.5. Adjusting Maximum heap size limit

   - Add the following parameters into the **Generic JVM arguments** field to set the perm space limit and direct memory limit:

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

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

   - Java runtime settings:

     Go to **Servers →Server Types →WebSphere application servers →[Server_Name] (default name: server1) →Java SDKs** and select version 1.7 as the default SDK.

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

3. Go to the download section of your [CloverETL account](#) and download the `clover.war` (web archive) file containing CloverETL Server for WebSphere.

4. Deploy the `clover.war` file.
   - Go to **Integrated Solutions Console** (default URL: http://localhost:9060/ibm/console/).
   - Go to **Applications** →**New Application** →**New Enterprise Application**, select the CloverETL Server WAR archive and deploy it to the application server, but do not start it yet.
5. Configure application class loading.

   Go to **WebSphere Enterprise Applications** →**clover_war (or other name of the Clover application)** →**Manage Modules** →**CloverETL** and under **Class loader order** select **Classes loaded with local class loader first (parent last)**.
6. Save the changes to the server configuration and start the **clover_war** application.
7. Check whether the server is running.

   Provided you set `clover.war` as the application running with "clover" context path, use the following URL (notice the port number has changed):

   http://localhost:9080/clover

Note

Please note that some CloverETL features using third party libraries do not 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.
- **AddressDoctor5** on IBM WebSphere requires additional JVM parameter `-Xmso2048k` to prevent AddressDoctor from crashing JVM. See documentation on AddressDoctor component.

Configuration of CloverETL Server on IBM WebSphere

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61,) or specified (p. 61) location.

Content of such a file (example with MySQL database):

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

Properties File in Specified Location

The properties file is loaded from a location which is specified by the environment/system property `clover_config_file` or `clover.config.file`.

1. Create the `cloverServer.properties` file in a directory readable by IBM WebSphere. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)

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

   a. Go to **Integrated Solutions Console** (default URL:http://localhost:9060/ibm/console/).

   b. Go to **Servers** →**WebSphere application servers** →**[Server_name]** →**Java and Process Management** →**Process Definition** →**Java Virtual Machine** →**Custom Properties**.

   c. Create system property named `clover_config_file` whose value is a full path to the properties file (e.g. `cloverServer.properties`) on your file system.

3. Restart IBM WebSphere for changes to take effect.

Note

➡**Continue with:** Chapter 4, Postinstallation Configuration (p. 55)

GlassFish / Sun Java System Application Server

Installation of GlassFish (p. 27)
Installation of CloverETL Server (p. 28)
Configuration of CloverETL Server on GlassFish (p. 29)

Important

See Application Server (p. 5) in system requirements for currently supported **GlassFish** versions and required **Java** versions.

GlassFish 3.1.2 contains a bug causing **Launch Services** to work improperly (see https://java.net/jira/browse/GLASSFISH-18444). We recommend version 3.1.2.2.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of GlassFish

1. Choose and download the GlassFish release from the official download page.

   CloverETL Server is developed and tested with the GlassFish 3.1.2.2 container. Running the Server with other versions may result in unpredictable behavior.

2. Extract the downloaded archive, or run the `.exe` file which will guide you through the setup.

3. Run GlassFish.

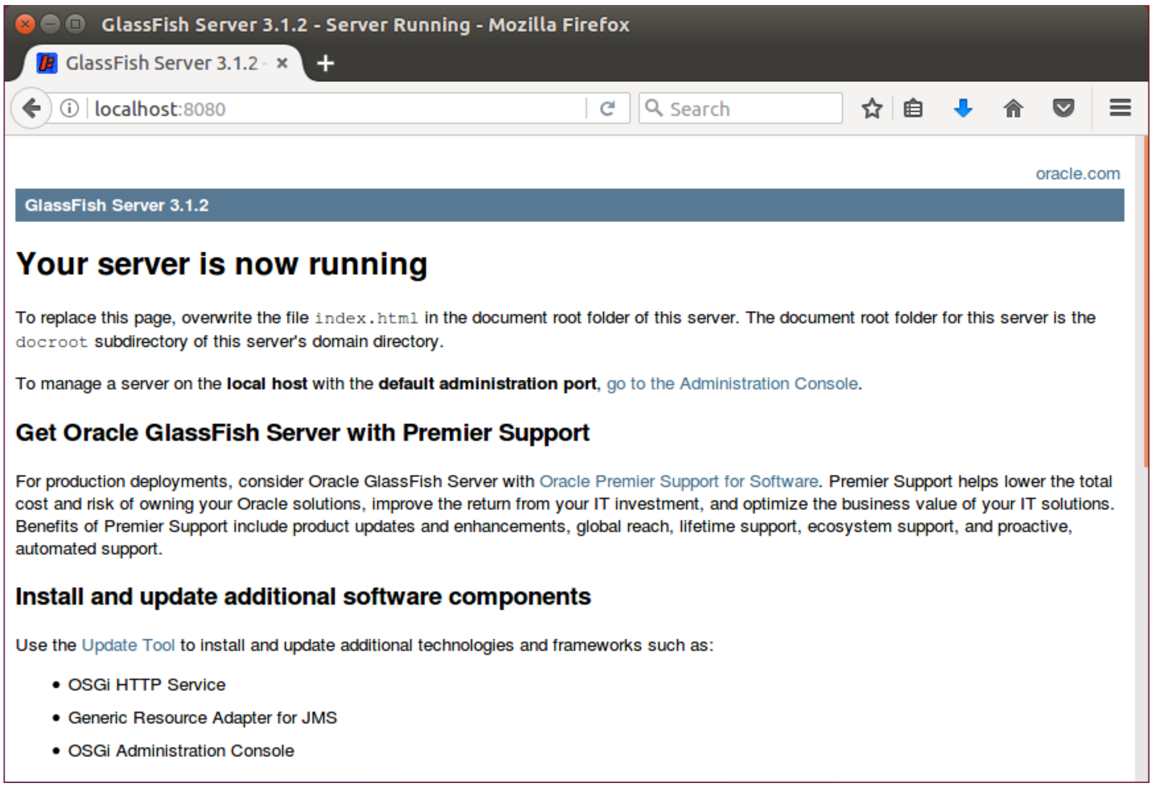
   - Run `[GlassFish_home]/bin/asadmin start-domain` and enter a new password for administrator.

4. Check whether GlassFish is running.

   - Open a new tab in your browser and type http://localhost:8080/ in the address bar.

     If the GlassFish welcome page appears (see below), the server is successfully installed.

   - The Admin Console is, by default, accessible at http://localhost:4848/.


GlassFish Server 3.1.2 - Server Running - Mozilla Firefox

localhost:8080

oracle.com

GlassFish Server 3.1.2

Your server is now running

To replace this page, overwrite the file `index.html` in the document root folder of this server. The document root folder for this server is the `docroot` subdirectory of this server's domain directory.

To manage a server on the **local host** with the **default administration port**, go to the Administration Console.

Get Oracle GlassFish Server with Premier Support

For production deployments, consider Oracle GlassFish Server with Oracle Premier Support for Software. Premier Support helps lower the total cost and risk of owning your Oracle solutions, improve the return from your IT investment, and optimize the business value of your IT solutions. Benefits of Premier Support include product updates and enhancements, global reach, lifetime support, ecosystem support, and proactive, automated support.

Install and update additional software components

Use the Update Tool to install and update additional technologies and frameworks such as:

- OSGi HTTP Service
- Generic Resource Adapter for JMS
- OSGi Administration Console

Figure 3.6. Glassfish welcome page

Installation of CloverETL Server

1. Check if you meet the prerequisites:
   - Oracle JDK or JRE is installed (see Java Virtual Machine (p. 5) for required java version).
   - **JAVA_HOME** or **JRE_HOME** environment variable is set.
   - A supported version (p. 6) of GlassFish 3 is installed.
2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters and **perm space** by adjusting the "XX:MaxPermSize" parameter:
   - Edit the `[GlassFish_home]/glassfish/domains/domain1/config/domain.xml` file.

     Change/add the following sub-elements in the `<java-config>` section:

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

   - Restart GlassFish.
3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for GlassFish 3.
4. Deploy the `clover.war` file.

- Open the **GlassFish Administration Console** (default URL: http://localhost:4848/).
- Go to **Applications** and click **Deploy ...**.
- Upload the `clover.war` file or select the file from a filesystem if it is present on the machine running GlassFish.
- Make sure the **Web Application** is selected in the **Type** field.

  Type "clover" in both the **Application Name** and the **Context Root** fields.
- Click **OK**.

Configuration of CloverETL Server on GlassFish

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61) or specified (p. 61) location.

Content of such a file (example with MySQL database):

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

Properties file in Specified Location

The properties file is loaded from a location which is specified by the environment/system property `clover_config_file` or `clover.config.file`.

1. Create the `cloverServer.properties` file in a directory readable by GlassFish. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)
2. Set system property `clover.config.file` pointing to the config properties file:
   a. Go to **GlassFish Administration Console** (default URL: http://localhost:4848/).
   b. Go to **Configuration** →**System Properties**
   c. Create system property named `clover.config.file` whose value is a full path to the file on your file system (e.g.: `/home/clover/cloverServer.properties`).
3. Copy the **JDBC driver** `.jar` file for a selected database into `[GlassFish_home]/glassfish/domains/[domain-name]/lib`
4. Restart GlassFish.

Note

Continue with: Chapter 4, Postinstallation Configuration (p. 55)

JBoss Application Server

Installation of JBoss AS (p. 30)
Installation of CloverETL Server (p. 31)
Configuration of CloverETL Server on JBoss AS (p. 33)

Important

See Application Server (p. 5) in system requirements for currently supported **JBoss AS** versions and required **Java** versions.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of JBoss AS

1. Download the JBoss AS release from the official download page.

   CloverETL Server is developed and tested with the JBoss AS 5, 6 and 7 containers. Running the Server with other versions may result in unpredictable behavior.

2. Extract the downloaded archive (`zip` or `tar.gz`).

3. Run JBoss AS.

   - **Unix-like systems:**
     - JBoss AS v 6.x.x

       `Run [JBoss_AS_home]/bin/run.sh`.
     - JBoss AS v 7.x.x

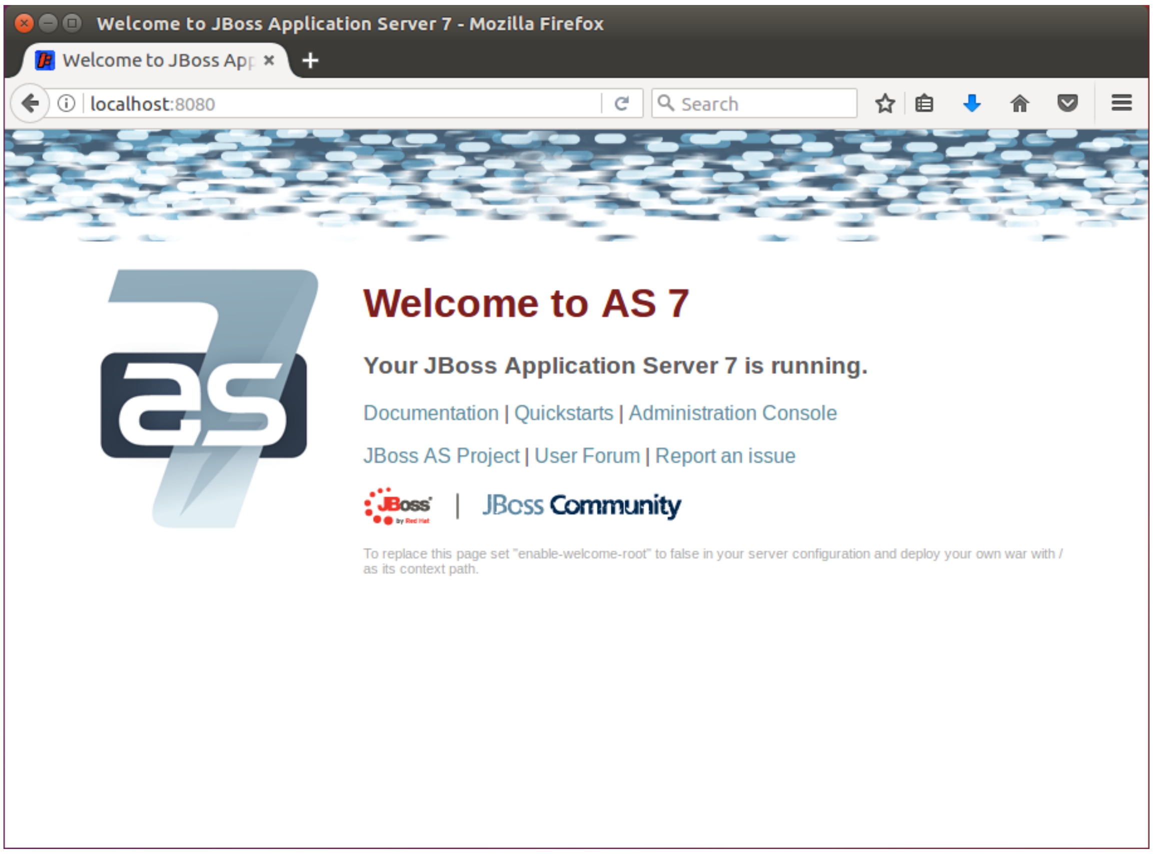
       `Run [JBoss_AS_home]/bin/standalone.sh`.
   - **Windows system:**
     - JBoss AS v 6.x.x

       `Run [JBoss_AS_home]\bin\run.bat`.
     - JBoss AS v 7.x.x

       `Run [JBoss_AS_home]\bin\standalone.bat`.

4. Check whether JBoss AS is running.

   - Open a new tab in your browser and type http://localhost:8080/ in the address bar.



Welcome to AS 7

Your JBoss Application Server 7 is running.

Documentation | Quickstarts | Administration Console

JBoss AS Project | User Forum | Report an issue

To replace this page set "enable-welcome-root" to false in your server configuration and deploy your own war with / as its context path.

Figure 3.7. JBoss AS welcome page

Installation of CloverETL Server

1. Check if you meet the prerequisites:
   - Oracle JDK or JRE is installed (see Java Virtual Machine (p. 5) for the required Java version).
   - **JAVA_HOME** or **JRE_HOME** environment variable is set.
   - A supported version (p. 6) of JBoss AS is installed.
2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters and **classloaders memory limit** by adjusting the "XX:MaxMetaspaceSize" parameter:

   - **Unix-like systems:**

JBoss AS v 6.x.x	Edit the `[JBoss_AS_home]/bin/run.conf`.
JBoss AS v 7.x.x	Edit the `[JBoss_AS_home]/bin/standalone.conf`.

     Edit the values of the following attributes:

```
-XX:MaxMetaspaceSize=512m -Xms128m -Xmx2048m"
```

   - **Windows system:**

JBoss AS v 6.x.x	Edit the `[JBoss_AS_home]\bin\run.conf.bat`.
JBoss AS v 7.x.x	Edit the `[JBoss_AS_home]\bin\standalone.conf.bat`.

Edit the values of the following attributes:

```
-XX:MaxMetaspaceSize=512m -Xms128m -Xmx2048m"
```

3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for JBoss AS.

4. Create a separate JBoss server configuration.
It may be useful to use a specific JBoss server configuration, when it is necessary to run CloverETL:

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

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

5. 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_AS_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, ampersand "&" as "&amp;" 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_AS_home]/server/[serverConfiguration]/lib`

6. Configure CloverETL Server according to the description in the next section (p. 33) .

7. Deploy the WAR file

   Copy `clover.war` to `[JBoss_AS_home]/server/[serverConfiguration]/deploy`

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

Configuration of CloverETL Server on JBoss AS

Note

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61,) or specified (p. 61) location.

Content of such a file (example with MySQL database):

```
datasource.type=JNDI
datasource.jndiName=java:/CloverETLServerDS
jdbc.dialect=org.hibernate.dialect.MySQLDialect
```

`datasource.type`	Indicates the server will use JNDI-bound datasource created in steps above.
`datasource.jndiName`	Specifies where can the datasource be found in JNDI.
`jdbc.dialect`	Set the dialect according to your database server (Part III, “Configuration” (p. 60)).

Properties File in Specified Location

The properties file is loaded from a location which is specified by the environment/system property `clover_config_file` or `clover.config.file`.

1. Create the `cloverServer.properties` file in a directory readable by JBoss AS. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)

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

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

   The simplest way is by setting a Java parameter:

   Unix-like systems:

   a. Edit the `[JBoss_AS_home]/bin/run.sh` file.

   b. Add the following line:

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

Windows system:

a. Edit the `[JBoss_AS_home]\bin\run.conf.bat` file.

b. Add the following line to the section where options are passed to the JVM:

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

Important

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

3. Restart JBoss AS so that the changes take effect.

4. Check the CloverETL Server application is running:

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

Note

Continue with: Chapter 4, Postinstallation Configuration (p. 55)

JBoss Enterprise Application Platform

Installation of JBoss EAP (p. 35)
Installation of CloverETL Server (p. 36)
Configuration of CloverETL Server on JBoss EAP (p. 39)

Important

See Application Server (p. 5) in system requirements for currently supported **JBoss EAP** versions and required **Java** versions.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of JBoss EAP

1. Using your credentials, log into the customer portal on the official Red Hat page and download a compatible JBoss EAP version.

2. Extract the downloaded archive (alternatively, you can download and run the installer and follow the instructions).

3. Run JBoss EAP.

   - **Unix-like systems:**

     Run `[JBoss_EAP_home]/bin/standalone.sh`.

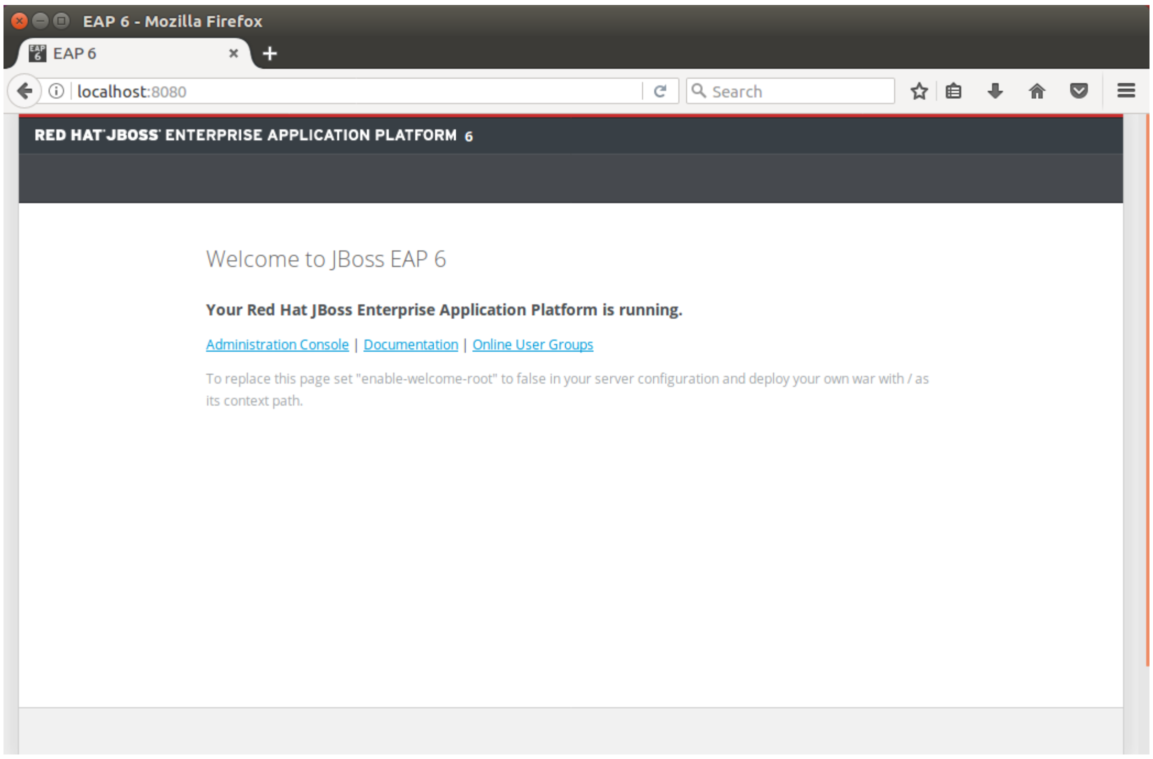
   - **Windows system:**

     Run `[JBoss_EAP_home]\bin\standalone.bat`.

4. Check whether JBoss EAP is running.

   - Open a new tab in your browser and type http://localhost:8080/ in the address bar.

     You should see the JBoss EAP welcome page (otherwise, please consult the JBoss EAP guide):



Welcome to JBoss EAP 6

Your Red Hat JBoss Enterprise Application Platform is running.

Administration Console | Documentation | Online User Groups

To replace this page set "enable-welcome-root" to false in your server configuration and deploy your own war with / as its context path.

Figure 3.8. JBoss EAP welcome page

Installation of CloverETL Server

1. Check if you meet the prerequisites:
   - Oracle JDK or JRE is installed (see Java Virtual Machine (p. 5) for the required Java version).
   - **JAVA_HOME** or **JRE_HOME** environment variable is set.
   - A supported version (p. 6) of JBoss EAP (JBoss AS 7) is installed.

     CloverETL Server is developed and tested with the JBoss EAP 6.2 and 6.4 (JBoss 7) containers. Running the Server with other versions may result in unpredictable behavior.
2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters and **classloaders memory limit** by adjusting the "XX:MaxMetaspaceSize" parameter:

   For JBoss EAP standalone mode, follow these steps:

   - **Unix-like systems:**
     - Edit the `[JBoss_EAP_home]/bin/standalone.conf` file.
     - Add the following line:

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

   - **Windows systems:**

- Edit the `[JBoss_EAP_home]\bin\standalone.conf.bat` file.

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

Important

If you use Java 7, change `-XX:MaxMetaspaceSize` to `-XX:MaxPermSize`.

3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for JBoss EAP.

4. Configure the database connection.

   By default, CloverETL Server uses embedded Derby database; however, such setup is not recommended for production use.

   You can use the database connection provided by JNDI-bound datasource deployed by JBoss EAP. In order to define the datasource, edit the file:

   `[JBoss_EAP_home]/standalone/configuration/standalone.xml`

   and add the definition of the datasourceinto into the section `<subsystem xmlns="urn:jboss:domain:datasources:1.1">` under the element `<datasources>`. Here is an example of datasource connecting to a 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>
```

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

Note

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, you need 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`):

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

b. 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>
```

c. Add the module to global server modules: in case of the standalone JBoss EAP server they are defined in `[JBoss_EAP_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 a description in the next section (p. 39).

7. Deploy WAR file.

   Copy the `clover.war` file to `[JBoss_EAP_home]/standalone/deployments.`

8. To start the JBoss platform:

   - **Unix-like systems:**

     Run `[JBoss_EAP_home]/bin/standalone.sh`.

   - **Windows system:**

     Run `[JBoss_EAP_home]\bin\standalone.bat`.

   It may take a couple of minutes for all applications to start.

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

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties(p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61) or specified (p. 61) location.

Properties File in Specified Location

The properties file is loaded from a location which is specified by the environment/system property `clover_config_file` or `clover.config.file`.

1. • Create the `cloverServer.properties` file in a directory readable by JBoss EAP. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).):

```
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. 60)). 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
```

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

   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_EAP_home]/standalone/configuration/standalone.xml`, and to add the following snippet just under `<extensions>` section:

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

3. Restart the JBoss EAP so that the changes take effect.

4. 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, disable the session replication. Edit `[jboss-home]/standalone/configuration/standalone.xml` and comment out whole `<cache-container name="web"  aliases="standard-session-cache">` block under `<subsystem xmlns="urn:jboss:domain:infinispan:1.5">` section.

Note

➡**Continue with:** Chapter 4, Postinstallation Configuration (p. 55)

Oracle WebLogic Server

Installation of Oracle Weblogic (p. 41)
Installation of CloverETL Server (p. 42)
Configuration of CloverETL Server on WebLogic (p. 43)

Important

See Application Server (p. 5) in system requirements for currently supported **Oracle WebLogic** versions and required **Java** versions.

If you encounter any problems during the installation, the Possible Issues during Installation (p. 50) section may provide a solution.

Installation of Oracle Weblogic

1. Create an Oracle account on http://www.oracle.com.

2. Go to the download page, accept the license agreement, download a compatible version of Weblogic server and extract the archive.

3. Set up a domain (the following steps are similar in Windows, simply run the `.cmd` files instead `.sh`).

   - Run [Weblogic_home]/configure.sh.

     After all files are unpacked and environment is set, you can configure your server domain. Alternatively, you can create and configure it in more details in the following step:

   - Run `[Weblogic_home]/oracle_common/bin/config.sh`.

     In the installer, you can create and configure your domain, administrator password and other parameters.

4. Start the server.

   - **Unix-like systems:**

     Run `[Weblogic_home]/user_projects/your_domain_name/startWeblogic.sh`.

   - **Windows system:**

     Run `[Weblogic_home]\user_projects\your_domain_name\startWeblogic.cmd`.

5. Launch the Administration Console (default URL: http://localhost:7001/console/).

   You should see the following welcome page:

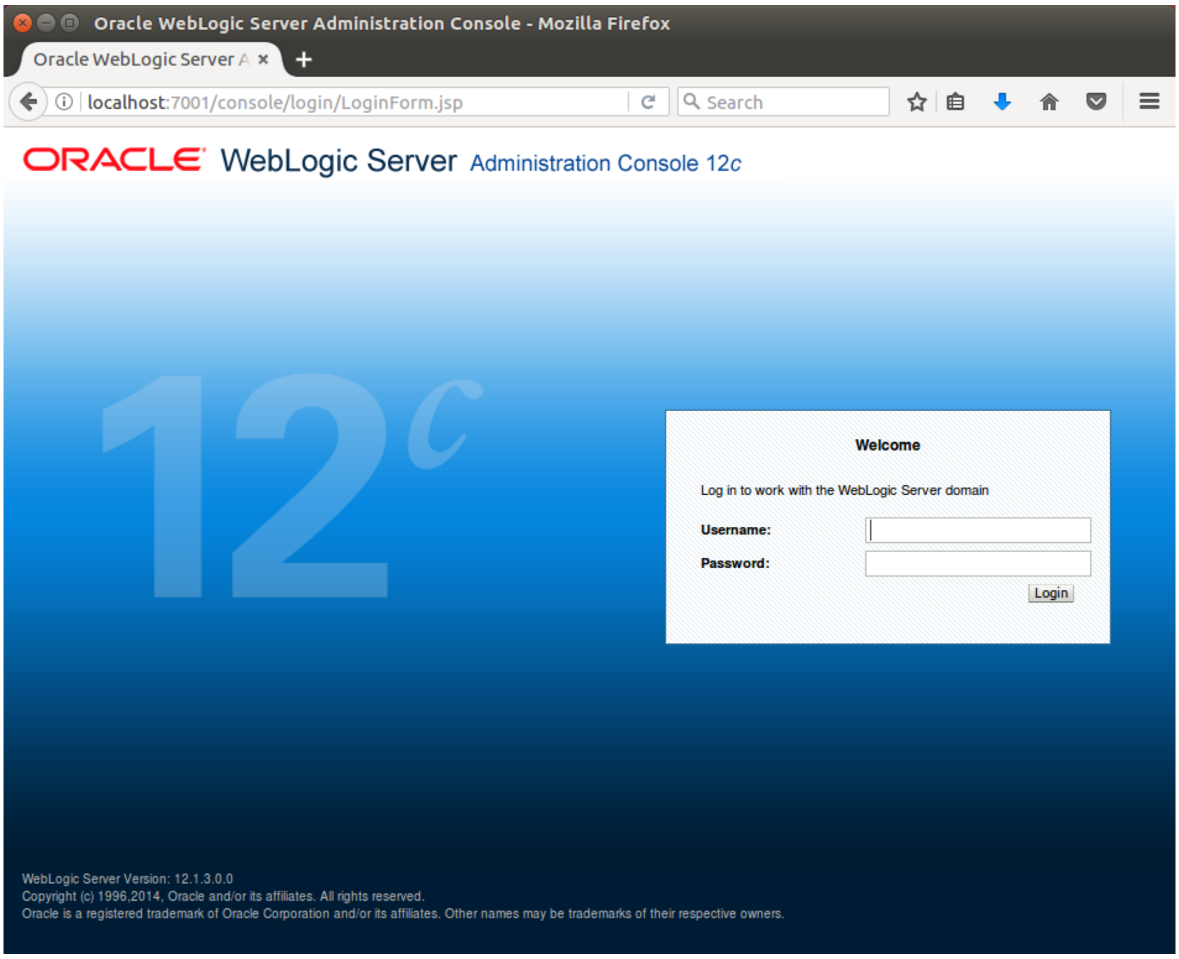

Oracle WebLogic Server Administration Console - Mozilla Firefox
Oracle WebLogic Server A
localhost:7001/console/login/LoginForm.jsp
Search
ORACLE WebLogic Server Administration Console 12c
12c
Welcome
Log in to work with the WebLogic Server domain
Username:
Password:
Login
WebLogic Server Version: 12.1.3.0.0
Copyright (c) 1996,2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Figure 3.9. WebLogic welcome page

Installation of CloverETL Server

1. Check if you meet the prerequisites:

   - Oracle JDK or JRE is installed (see Java Virtual Machine (p. 5) for the required Java version).
   - **JAVA_HOME** or **JRE_HOME** environment variable is set.
   - A supported version (p. 6) of Oracle WebLogic Server is installed.

     CloverETL Server is developed and tested with the WebLogic Server 11g (10.3.6) and WebLogic Server 12c (12.1.2) containers. Running the Server with other versions may result in unpredictable behavior.

     WebLogic has to be running and a domain has to be configured. You can check it by connecting to **Administration Console**: http://localhost:7001/console/. **Username** and **password** are specified during installation.

2. It is strongly recommended to adjust the default limits for **Memory allocation** (see the Memory Settings (p. 55) section).

   You can set the **minimum** and **maximum memory heap size** by adjusting the "Xms" and "Xmx" JVM parameters and **classloaders memory limit** by adjusting the "XX:MaxMetaspaceSize" parameter:

   - **Unix-like systems:**

     Edit the start script and add:

```
export JAVA_OPTIONS='$JAVA_OPTIONS -Xms512m -Xmx2048m -XX:MaxMetaspaceSize=512m'
```

- **Windows system:**

  See WebLogic Server Performance and Tuning.

Important

If you use Java 7, change `-XX:MaxMetaspaceSize` to `-XX:MaxPermSize`.

3. Go to the download section of your CloverETL account and download the `clover.war` (web archive) file containing CloverETL Server for Oracle WebLogic Server.

4. 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 authenticate users itself.

   - Edit the config file `[domainHome]/config/config.xml` and add:

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

   into `<security-configuration>` element (just before the end tag).

5. Deploy `clover.war` (or an application directory).

   Use the **WebLogic Server Administration Console**. See the Oracle Fusion Middleware Administrator's Guide for details.

6. Configure a license and other properties. See Configuration of CloverETL Server on WebLogic (p. 43) for details.

7. Check whether CloverETL Server is running.

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

   - CloverETL Server is accessible at http://host:7001/clover by default.

Configuration of CloverETL Server on WebLogic

Tip

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

For detailed configuration of CloverETL Server, use a **properties file**. Here you can configure various properties, including the connection to the database, username and password, path to the license file, private properties, number of active threads, clusters and much more (see Chapter 9, List of Properties (p. 88) and Chapter 30, Cluster Configuration (p. 260)). The file can be placed either on a default (p. 61) or specified (p. 61) location.

Content of such a file (example with MySQL database):

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

Properties File in Specified Location

1. Create the `cloverServer.properties` file in a directory readable by WebLogic. (If you need an example of connection to any of supported databases, see Chapter 8, Examples of DB Connection Configuration (p. 71).)

   Config file should contain DB datasource config, SMTP connection config, etc. See Part III, “Configuration” (p. 60) for details.

2. Set `clover_config_file` system property (or environment variable) 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
```

3. Restart WebLogic for changes to take effect.

Important

When CloverETL Server is deployed on WebLogic and JNDI Datasource pointing to Oracle DB is used, there must be an extra config property in the config file:

```
quartz.driverDelegateClass=org.quartz.impl.jdbcjobstore.oracle.weblogic.WebLogicOracleDelegate
```

Note

Continue with: Chapter 4, Postinstallation Configuration (p. 55)

CloverETL Server Activation

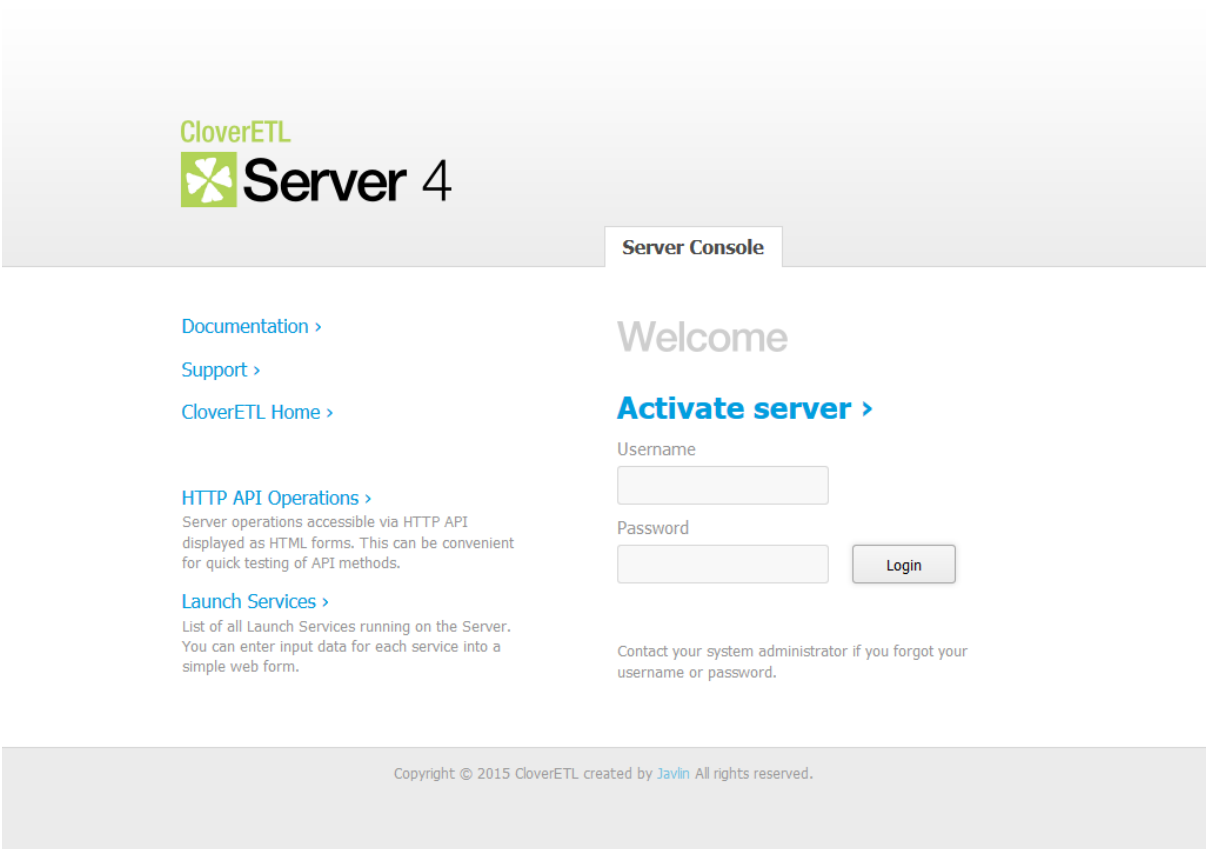
To be able to execute graphs, CloverETL Server requires a valid license. You can install and run CloverETL Server without any license, but no graph will be executed.

There are three ways of installing the license. They work on all application servers and can be used at the same time, but **only the most recent valid license is used**.

We recommend using the first and easiest option (for other options see CloverETL Server Activation Alternatives (p. 47)):

CloverETL Server Activation using 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.



Server operations accessible via HTTP API displayed as HTML forms. This can be convenient for quick testing of API methods.

List of all Launch Services running on the Server. You can enter input data for each service into a simple web form.

Contact your system administrator if you forgot your username or password.

Copyright © 2015 CloverETL created by Javlin All rights reserved.

Figure 3.10. Login page of CloverETL Server without license

You can paste a 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.

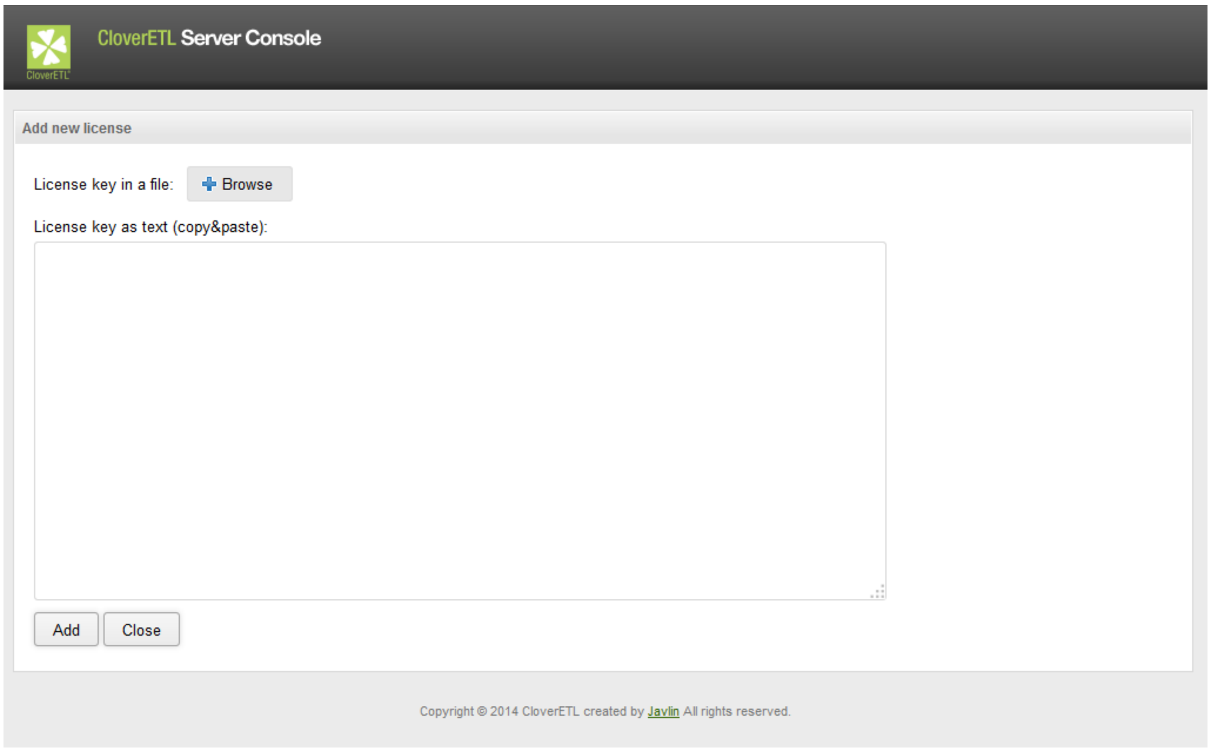

CloverETL Server Console
Add new license
License key in a file:
Browse
License key as text (copy&paste):
Add
Close
Copyright © 2014 CloverETL created by Javlin All rights reserved.

Figure 3.11. Add new license form

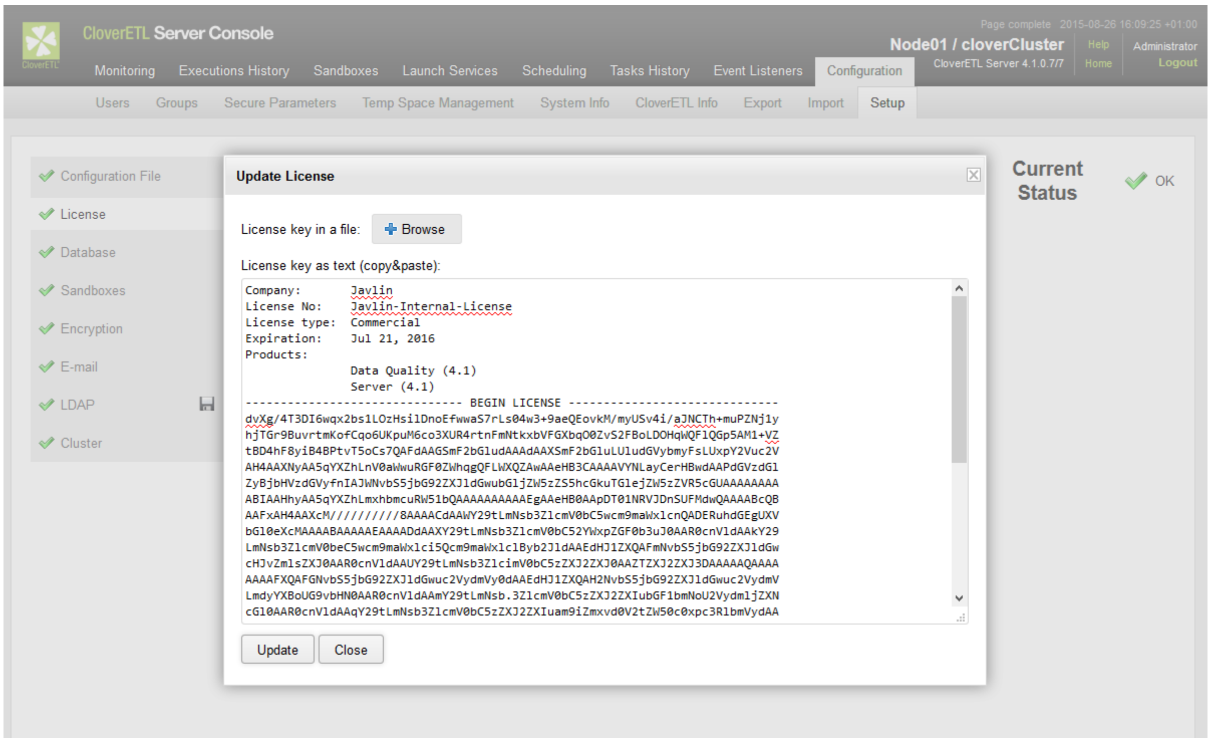
Updating CloverETL Server License in the Configuration Section

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

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



Update License

License key in a file:

License key as text (copy&paste):

```
Company:       Javlin
License No:    Javlin-Internal-License
License type:  Commercial
Expiration:    Jul 21, 2016
Products:
               Data Quality (4.1)
               Server (4.1)
------------------------------ BEGIN LICENSE ------------------------------
dvXg/4T3DI6wqx2bs1LOzHsilDnoEfwwaS7rLs04w3+9aeQEovkM/myUSv4i/aJNCTh+muPZNj1y
hjTGr9BuvrtmKofCqo6UKpuM6co3XUR4rtnFmNtkxbVFGXbqO0ZvS2FBoLDOHqWQF1QGp5AM1+VZ
tBD4hF8yiB4BPtvT5oCs7QAFdAAGSmF2bGludAAAdAAXSmF2bGluLUludGVybmyFsLUxpY2Vuc2V
AH4AAXNyAA5qYXZhLnV0aWwuRGF0ZWhqgQFLWXQZAwAAeHB3CAAAAVYNLayCerHBwdAAPdGVzdGl
ZyBjbHVzdGVyfnIAJWNvbS5jbG92ZXJ1dGwubGljZW5zZS5hcGkuTGlejZW5zZVR5cGUAAAAAAAA
ABIAAHhyAA5qYXZhLmxhbmcuRW51bQAAAAAAAAAAEgAAeHB0AApDT01NRVJDnSUFMdwQAAAABcQB
AAFxAH4AAXcM//////////8AAAACdAAWY29tLmNsb3ZlcmV0bC5wcm9maWxlcnQADERuhdGEgUXV
bGl0eXcMAAAABAAAAAEAAAADdAAXY29tLmNsb3ZlcmV0bC52YWxpZGF0b3uJ0AAR0cnVldAAkY29
LmNsb3ZlcmV0beC5wcm9maWxlci5Qcm9maWxlclByb2J1dAAEdHJ1ZXQAFmNvbS5jbG92ZXJ1dGw
cHJvZmlsZXJ0AAR0cnVldAAUY29tLmNsb3ZlcimV0bC5zZXJ2ZXJ0AAZTZXJ2ZXJ3DAAAAAQAAAA
AAAAFXQAFGNvbS5jbG92ZXJ1dGwuc2VydmVy0dAAEdHJ1ZXQAH2NvbS5jbG92ZXJ1dGwuc2VydmV
LmdyYXBoUG9vbHN0AAR0cnVldAAmY29tLmNsb.3ZlcmV0bC5zZXJ2ZXIubGF1bmNoU2VydmljZXN
cGl0AAR0cnVldAAqY29tLmNsb3ZlcmV0bC5zZXJ2ZXIuam9iZmxvd0V2tZW50c0xpc3RlbmVydAA
```

Figure 3.12. Update license form

Tip

CloverETL license can be **changed** at 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

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

Continue with: Chapter 7, Setup (p. 63)

CloverETL Server Activation Alternatives

If, for any reason, you decide to not use the recommended way of installing the server license, you can choose one of the following options:

Activation Using 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. 88) 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 `clover-license.war` web archive file.
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: `clover-license` contextPath is mandatory and cannot by changed)

Note

➡**Continue with:** Chapter 7, Setup (p. 63)

IBM InfoSphere MDM Plugin Installation

Downloading

IBM InfoSphere 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 `ibm-mdm-connectors.${version}.zip` file.

Requirements

Requirements of **IBM InfoSphere MDM Components**:

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

The support for 32 bit Linux was removed in 4.5.0.

Installation into Server

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

1. Install **CloverETL Server**, see its documentation for details.
2. Download the ZIP file with **IBM InfoSphere MDM Components** for Server and store it on the system where **CloverETL Server** is installed. See Downloading (p. 48) 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/ibm-mdm-connectors.4.5.0.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 IBM InfoSphere MDM components, it means that the **IBM InfoSphere 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 with a proper configuration of the server environment. This section contains tips for the configuration.

Memory Issues on Derby (p. 50)
JAVA_HOME or JRE_HOME Environment Variables Are Not Defined (p. 50)
Apache Tomcat Context Parameters Do Not Have Any Effect (p. 51)
Tomcat Log File catalina.out Is Missing on Windows (p. 51)
Timeouts Waiting for JVM (p. 51)
clover.war as Default Context on WebSphere (Windows OS) (p. 52)
Tomcat 6.0 on Linux - Default DB (p. 52)
Derby.system.home Cannot be Accessed (p. 52)
Environment Variables and More than one CloverETL Server Instances Running on Single Machine (p. 53)
Special Characters and Slashes in Path (p. 53)
File System Permissions (p. 53)
JMS API and JMS Third-Party Libraries (p. 53)
Using an Unsupported JDBC Connector for MySQL (p. 53)

Memory Issues on Derby

If your server suddenly starts consuming too much resources (CPU, memory) despite having been working well before, it might be because 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 the config file, if you configured any.
3. Delete the `webapps/clover` directory.
4. Start Apache Tomcat server. It will automatically redeploy CloverETL Server.
5. Verify you can connect from Designer and from web.
6. Shutdown Apache Tomcat.
7. Restore the 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:

This command will help you set a path to the variable on the server.
```
[root@server /] export JAVA_HOME=/usr/local/jdk1.x.x
```

As a final step, restart the application server.

Windows OS:

Set `JAVA_HOME` to your JDK installation directory, e.g. `C:\Program Files\java\jdk1.8.0`.

Important

Some CloverETL functions requires JDK to work correctly, therefore we do not recommend having only JRE installed.

Apache Tomcat Context Parameters Do Not Have Any Effect

Tomcat may sometimes ignore some context parameters. It may cause weird CloverETL Server behavior, since it appears as configured, but only partially. Some parameters are accepted, some are ignored. This issue is rare, however it may occur in some environments. Such behavior is consistent, so restart has no effect. It's possibly 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 a 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. The `catalina.out` file may be vital when Tomcat isn't started in the console and an 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 the lines where the "_EXECJAVA" variable is set. There should be two such lines:

  `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 a new startup file, e.g. [Tomcat_home]/bin/startupLog.bat, containing a single 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 the 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 CloverETL 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 following line:

```
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
```

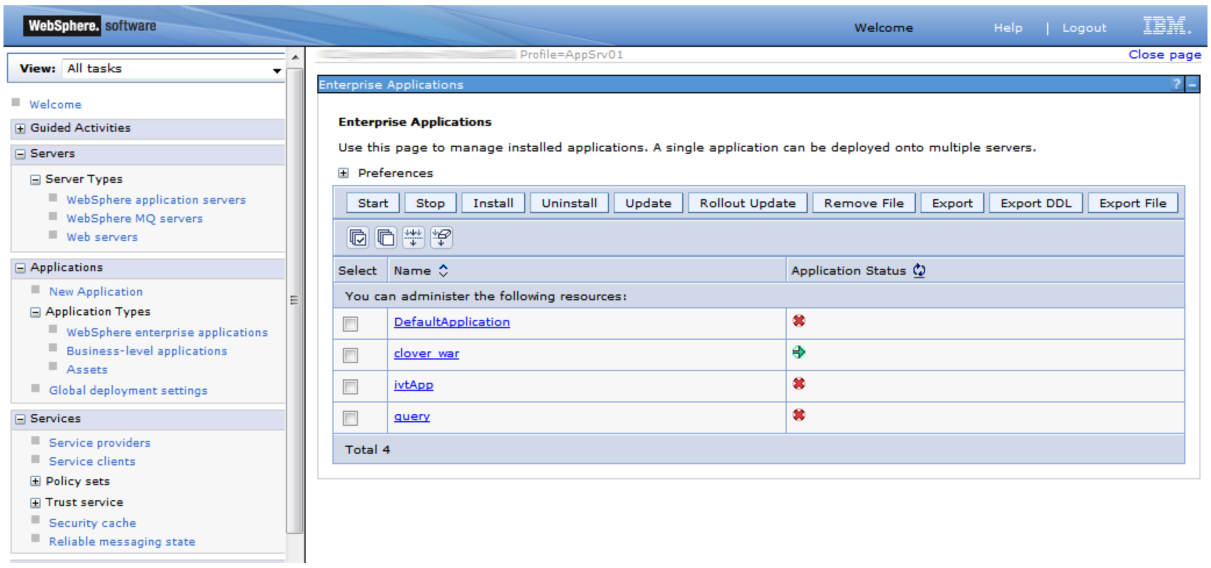
If that does not help either, try setting 120 for both values. Default timeouts are 30.

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 CloverETL Server on WebSphere, check the log and look for a following message:

```
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. The easiest way to fix the issue 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/).



Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

Select	Name	Application Status
You can administer the following resources:		
☐	DefaultApplication	✖
☐	clover_war	➔
☐	ivtApp	✖
☐	query	✖
Total 4		

Figure 3.13. CloverETL 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 CloverETL Server might fail on the 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 CloverETL Server deployed as a `WAR` file via the Tomcat web administration tool.

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 need to run more instances at once, use other ways of setting parameters (see Part III, "Configuration" (p. 60) for description of all possibilities). The reason is the environment variables are shared by all applications in use causing them to share configurations and fail unexpectedly. Instead of the environment variables, you can use system properties (passed to the application container process using parameter with -D prefix: -Dclover_config_file).

Special Characters and Slashes 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 not recommended. It's unfortunately 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, install the application server in a safe destination like:

```
C:\JBoss6\
```

Similarly, use slashes but never backslashes in paths inside the `*.properties` files, e.g. when pointing to the CloverETL Server license file. If you incorrectly use backlash, it will be considered an escape character and the server may not work properly. 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 with 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 the server startup, but it is an issue of deployment on an application server, thus it is still related to this chapter.

clover.war itself does not contain jms.jar, so it has to be on an application server's classpath. Most of the application servers have jms.jar by default, but Tomcat, for example, 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 a 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 an exception, for example:

```
could not execute query
You have an error in your SQL syntax; check the manual that corresponds 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 **optimal 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 is exhausted, the OutOfMemoryError is raised.

JVM Memory Areas

JVM memory consists of several areas: **heap memory**, **PermGen space**, **direct memory** and **stack memory**. Since JVM memory is not just HEAP memory, you should not set the HEAP limit too high; in case it consumes whole RAM, JVM won't be able to allocate direct memory and stack for new threads.

Table 4.1. JVM Memory Structure

Type	Description
Heap memory	Heap is an area of memory used by JVM for dynamic memory allocation. Required heap memory size depends on various factors (e.g. complexity of graphs, number of graphs running in parallel, type of component, etc.), see respective server container's installation guide in this documentation. (Note that current heap memory usage can be observed in CloverETL Server Console (p. 142).)
PermGen Space	Permanent Generation - separate memory space containing class definitions and related metadata. (PermGen was removed from Java 8.)
Direct Memory	Memory used by graph edges and buffers for I/O operations.
Stack Memory	Stack Memory contains local, method specific variables and references to other objects in the method. Each thread has its own stack; therefore, the memory usage depends on the number of components running in parallel.

Configuring Memory

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

Recommended Heap Memory Configuration

If you are not sure about the memory requirements for the transformations, a **maximum of 1-2 GB heap memory is recommended**. This limit may be increased during transformations development when `OutOfMemoryError` occurs.

Heap limit is *not* a limit of memory used by JVM. If you do not understand JVM in details, you should not assign more than 50% of main memory to heap.

Memory Configuration in Different Java Versions

In Java 7 and earlier, the memory space for loading classes (so called "PermGen space") is separated from the 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 enough in most cases. If the PermGen space maximum is too low, `OutOfMemoryError: PermGen space` may occur.

In Java 8, memory space for loading classes (so called "Metaspace") is separated from heap, and can be set by the JVM parameter -XX:MaxMetaspaceSize. The default maximum Metaspace size is unlimited.

Please see the specific container section for details on memory settings.

Codecache Size

Some CloverETL Server installations can occasionally run into performance issue: JVM is running more than hundred times slower. The issue can be caused by a full code cache (https://docs.oracle.com/javase/8/embedded/develop-apps-platforms/codecache.htm). Reserved code cache size is platform dependent and can be too small for CloverETL Server. It is highly recommended to increase code cache size using the following JVM argument:

```
-XX:ReservedCodeCacheSize=256m
```

Maximum Number of Open Files

When using resource-demanding components, such as FastSort, or when running a large number of graphs concurrently, you may reach the system limit on 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.

Firewall Exceptions

In order to function properly, CloverETL Server requires an outside communication. The table below describes both incoming and outgoing communication of CloverETL Server. Please, configure your firewall exceptions accordingly.

Table 4.2. Firewall Exceptions

Traffic	Communicat	Description & Components
Incoming	HTTP(S)	Communication between Designer and Server
	JMX	Tracking and debugging information
Outgoing (depending on an actual usage)	JDBC	connection to databases (DBInputTable, DBOutputTable, DBExecute)
	JMX	(JMSReader, JMSWriter, JMS Listener)
	HTTP(S)	(Readers, WebserviceClient, HTTPConnector)
	SMTP	(EmailSender)
	IMAP/ POP3	(EmailReader)
	FTP/ SFTP/ FTPS:	(readers, writers)

Note

Continue with: Chapter 8, Examples of DB Connection Configuration (p. 71)

Chapter 5. Upgrading Server to Newer Version

General Notes on Upgrade

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

Upgrade Prerequisites

- Having a new CloverETL Server web application archive (`clover.war` appropriate for the application server used) & license files available.
- Having release notes for the particular CloverETL version available (and all versions between current and intended version to be upgraded to).
- Having the graphs and jobs updated and tested with regards to Known Issues & Compatibility for the particular CloverETL version.
- Having the CloverETL Server configuration properties file externalized from default location, see Chapter 6, Configuration Sources and Their Priorities (p. 61).
- Standalone database schema where CloverETL Server stores configuration, see Chapter 8, Examples of DB Connection Configuration (p. 71).
- Having a separate sandbox with a test graph that can be run at any time 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 the CloverETL Server application (or all servers, if they run in a cluster mode).
3. Backup the 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 the existing CloverETL web application archive (`clover.war`) & license files (on all nodes).
5. Backup the existing CloverETL sandboxes (on all nodes).
6. Re-deploy the CloverETL Server web application. Instructions how to do that are application server dependent - see Production Server (p. 12) for installation details on all supported application servers. After the re-deployment, 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 the 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 license 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 CloverETL Server Activation (p. 45) for details on license installation.
8. Start the CloverETL Server application (on all nodes).

9. Review that contents of all tabs in the 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 the CloverETL Server application.
2. Restore the CloverETL Server web application (`clover.war`) & license files (on all nodes).
3. Restore the CloverETL Server database schema.
4. Restore the CloverETL sandboxes (on all nodes).
5. Start the 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 production server, you should not use the default configuration and database. Instead, take some time to configure CloverETL Server so that it best fits your production environment.

Part III. Configuration

This part describes in detail the configuration options for CloverETL Server used in production. In the following chapters, you will find information on setting required properties and parameters, description of CloverETL Server's Configuration GUI elements, parameters for specific database configuration, list of properties used in general configuration, instructions on encrypting confidential properties and log files setting.

Note

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. Configuration Sources and Their Priorities

Configuration Sources

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

Configuration properties can be encrypted (see details in Chapter 10, Secure Configuration Properties (p. 94)).

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

Environment Variables

Environment variables are variables configured by means of your operating system. E.g. `$PATH` is an environment variable.

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

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

System Properties

System properties are configured by means of JVM, e.g. with `-D` argument (`-Dclover.config.file`).

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

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

Properties File on Default Location

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

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

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

Properties File on Specified Location

A file has the same file structure as in the case above, but its location is specified in the `clover_config_file` or with a `clover.config.file` environment variable or system property.

This is the recommended way of configuration if context parameters cannot be set in the application server.

Modification of Context Parameters in web.xml

Unzip `clover.war` and modify the `WEB-INF/web.xml` file. Add the following piece of code into the file:

```
<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 the approaches above are possible.

Context Parameters (Available on Apache Tomcat)

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

This way of configuration is possible, but it is not recommended, as Apache Tomcat may ignore some context parameters in some environments. Use of a 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 a context configuration file `[Tomcat_home]/conf/Catalina/localhost/clover.xml` which is created automatically just after deployment of a CloverETL Server web application.

You can specify a property by adding this element:

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

Priorities of Configuration Sources

Configuration sources have these priorities:

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

Note

Continue with: Chapter 8, Examples of DB Connection Configuration (p. 71)

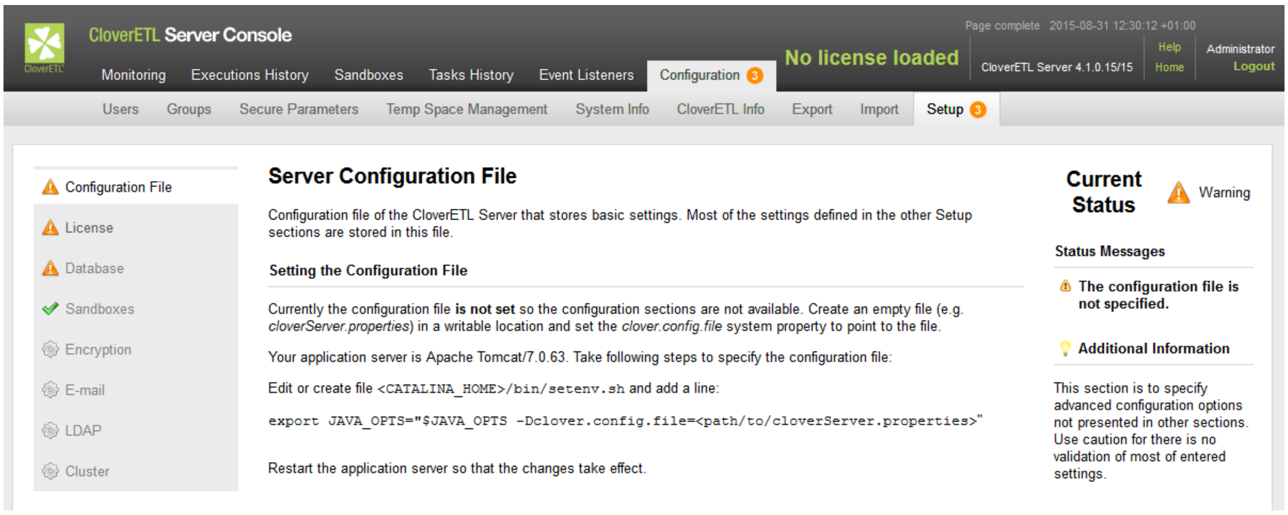
Chapter 7. Setup

The **CloverETL Server Setup** assists you with configuration of CloverETL Server. Instead of typing the whole configuration file in a text editor, the **Setup** generates the content of the configuration file according to your instructions. It lets 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 a server without configuration, you can see decorators pointing to the Setup. The decorators mark problems to be solved. The displayed number corresponds to the number of items.



Server Configuration File

Configuration file of the CloverETL Server that stores basic settings. Most of the settings defined in the other Setup sections are stored in this file.

Setting the Configuration File

Currently the configuration file **is not set** so the configuration sections are not available. Create an empty file (e.g. *cloverServer.properties*) in a writable location and set the *clover.config.file* system property to point to the file.

Your application server is Apache Tomcat/7.0.63. Take following steps to specify the configuration file:

Edit or create file `<CATALINA_HOME>/bin/setenv.sh` and add a line:

```
export JAVA_OPTS="$JAVA_OPTS -Dclover.config.file=<path/to/cloverServer.properties>"
```

Restart the application server so that the changes take effect.

Current Status Warning

Status Messages

The configuration file is not specified.

Additional Information

This section is to specify advanced configuration options not presented in other sections. Use caution for there is no validation of most of entered settings.

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

- ❌ error
- ⚠ warning
- 🔄 restart required

The **Setup** will help you with solving the problems.

Path to the Configuration File

Firstly, you have to specify a path to a configuration file. Without this, the Setup does not know, to which file the configuration should be saved. Each application server has a 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. 13).

Jetty

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

See also Jetty (p. 19).

GlassFish

Add the `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. 27).

JBoss

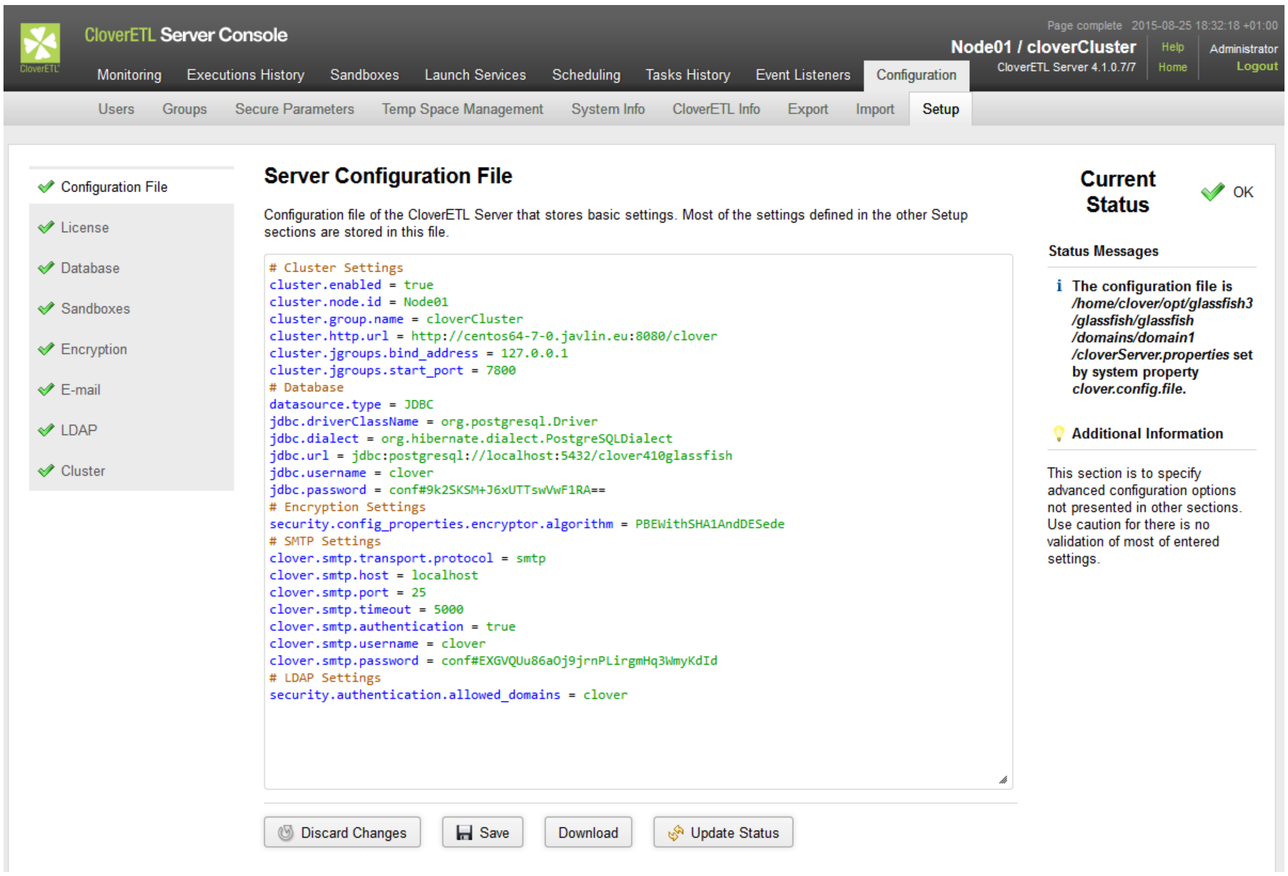
See also JBoss Application Server (p. 30).

WebSphere

See also IBM WebSphere (p. 23).

WebLogic

See also Oracle WebLogic Server (p. 41).



Server Configuration File

Configuration file of the CloverETL Server that stores basic settings. Most of the settings defined in the other Setup sections are stored in this file.

```
# Cluster Settings
cluster.enabled = true
cluster.node.id = Node01
cluster.group.name = cloverCluster
cluster.http.url = http://centos64-7-0.javlin.eu:8080/clover
cluster.jgroups.bind_address = 127.0.0.1
cluster.jgroups.start_port = 7800
# Database
datasource.type = JDBC
jdbc.driverClassName = org.postgresql.Driver
jdbc.dialect = org.hibernate.dialect.PostgreSQLDialect
jdbc.url = jdbc:postgresql://localhost:5432/clover410glassfish
jdbc.username = clover
jdbc.password = conf#9k2SKSM+J6xUTTswVwF1RA==
# Encryption Settings
security.config_properties.encryptor.algorithm = PBEWithSHA1AndDESede
# SMTP Settings
clover.smtp.transport.protocol = smtp
clover.smtp.host = localhost
clover.smtp.port = 25
clover.smtp.timeout = 5000
clover.smtp.authentication = true
clover.smtp.username = clover
clover.smtp.password = conf#EXGVQUu86aOj9jrnPLirgmHq3WmyKdId
# LDAP Settings
security.authentication.allowed_domains = clover
```

Current Status OK

Status Messages

The configuration file is *home/clover/opt/glassfish3/glassfish/glassfish/domains/domain1/cloverServer.properties* set by system property *clover.config.file*.

Additional Information

This section is to specify advanced configuration options not presented in other sections. Use caution for there is no validation of most of entered settings.

Adding Libraries to Classpath

Secondly, you should configure a connection to a database.

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

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

Configuring Particular Items

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

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

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

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

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

Setup Tabs

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

The menu tabs have **icons** surrounding the text: a **tick** marks a configured tab, a **wheel** marks an inactive tab, a **floppy disk** marks a configuration that needs saving. **Arrows** signalize a request on restart.

The main configuration part contains several buttons:

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

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

Validate validates the configuration on a current tab.

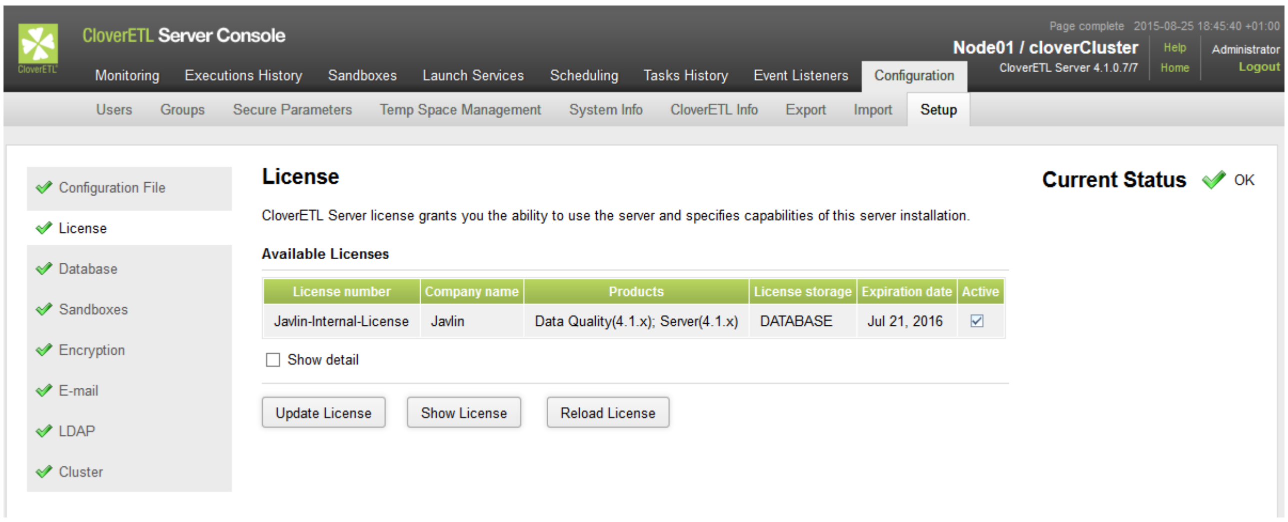
Configuration File

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

License

The **License** tab lets you specify the license. The license is stored in database.

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



License

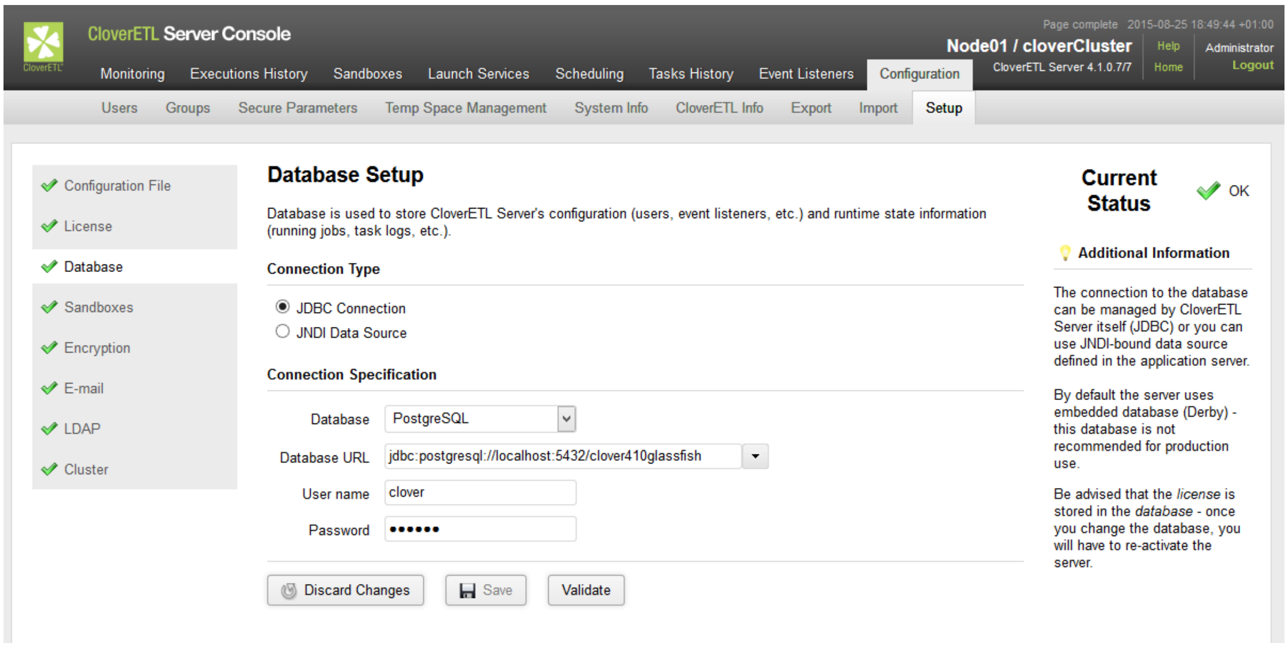
CloverETL Server license grants you the ability to use the server and specifies capabilities of this server installation.

Available Licenses

License number	Company name	Products	License storage	Expiration date	Active
Javlin-Internal-License	Javlin	Data Quality(4.1.x); Server(4.1.x)	DATABASE	Jul 21, 2016	☑

Database

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



Database Setup

Database is used to store CloverETL Server's configuration (users, event listeners, etc.) and runtime state information (running jobs, task logs, etc.).

Connection Type

(●) JDBC Connection
( ) JNDI Data Source

Connection Specification

Database	PostgreSQL
Database URL	jdbc:postgresql://localhost:5432/clover410glassfish
User name	clover
Password	••••••

Current Status ✓ OK

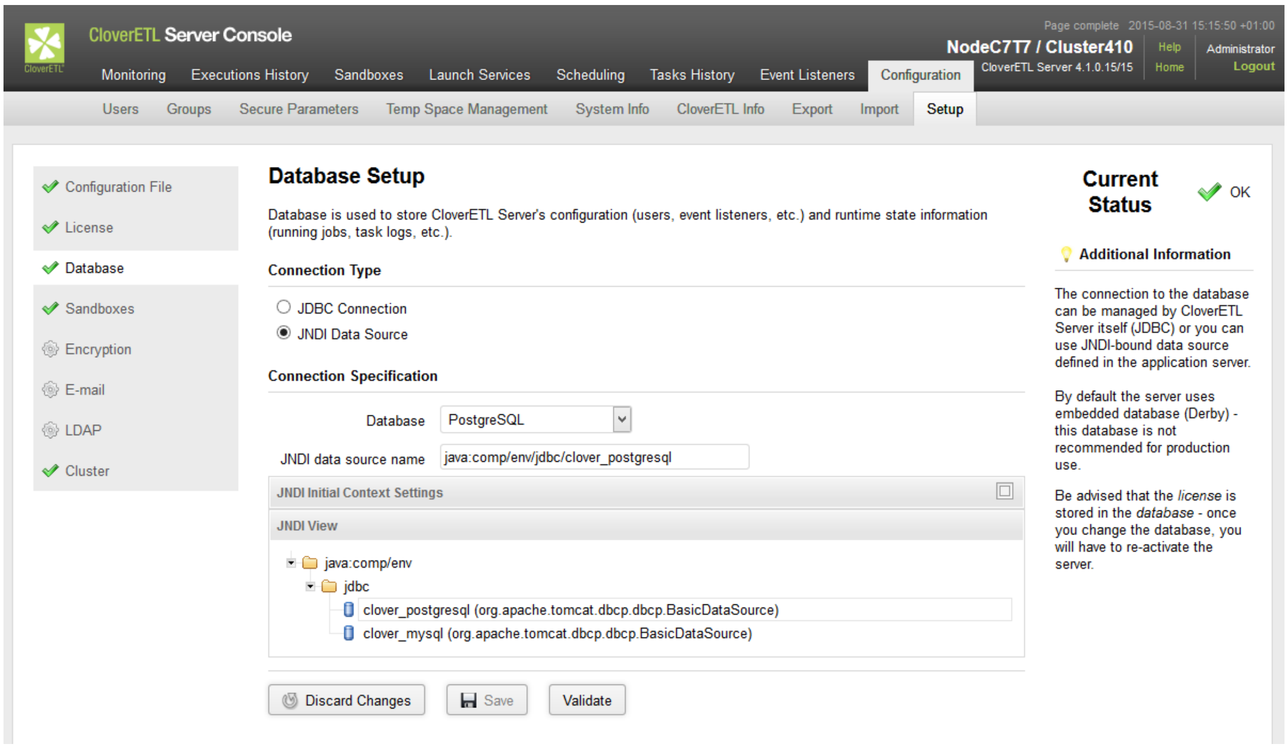
Additional Information

The connection to the database can be managed by CloverETL Server itself (JDBC) or you can use JNDI-bound data source defined in the application server.

By default the server uses embedded database (Derby) - this database is not recommended for production use.

Be advised that the *license* is stored in the *database* - once you change the database, you will have to re-activate the server.

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



Database Setup

Database is used to store CloverETL Server's configuration (users, event listeners, etc.) and runtime state information (running jobs, task logs, etc.).

Connection Type

( ) JDBC Connection
(●) JNDI Data Source

Connection Specification

Database	PostgreSQL
JNDI data source name	java:comp/env/jdbc/clover_postgresql

JNDI Initial Context Settings

JNDI View

```
java:comp/env
  jdbc
    clover_postgresql (org.apache.tomcat.dbcp.dbcp.BasicDataSource)
    clover_mysql (org.apache.tomcat.dbcp.dbcp.BasicDataSource)
```

Current Status ✓ OK

Additional Information

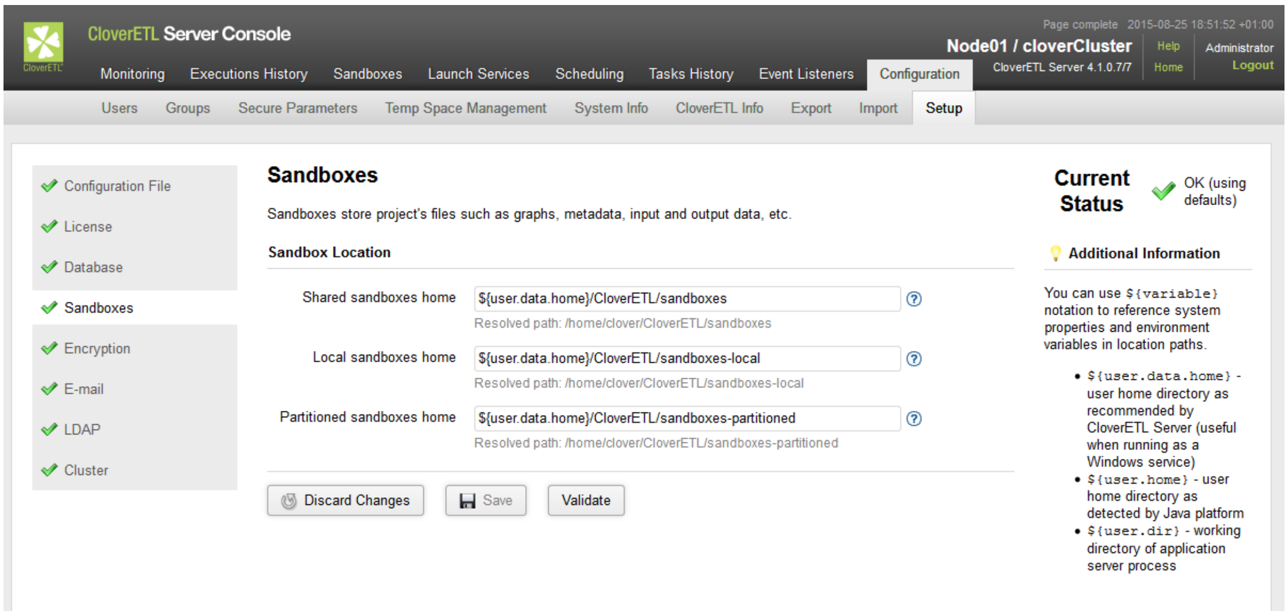
The connection to the database can be managed by CloverETL Server itself (JDBC) or you can use JNDI-bound data source defined in the application server.

By default the server uses embedded database (Derby) - this database is not recommended for production use.

Be advised that the *license* is stored in the *database* - once you change the database, you will have to re-activate the server.

Sandboxes

The **Sandboxes** tab lets you configure a path to sandboxes: shared, local, partitioned.



Sandboxes

Sandboxes store project's files such as graphs, metadata, input and output data, etc.

Sandbox Location

Shared sandboxes home	${user.data.home}/CloverETL/sandboxes Resolved path: /home/clover/CloverETL/sandboxes
Local sandboxes home	${user.data.home}/CloverETL/sandboxes-local Resolved path: /home/clover/CloverETL/sandboxes-local
Partitioned sandboxes home	${user.data.home}/CloverETL/sandboxes-partitioned Resolved path: /home/clover/CloverETL/sandboxes-partitioned

Current Status OK (using defaults)

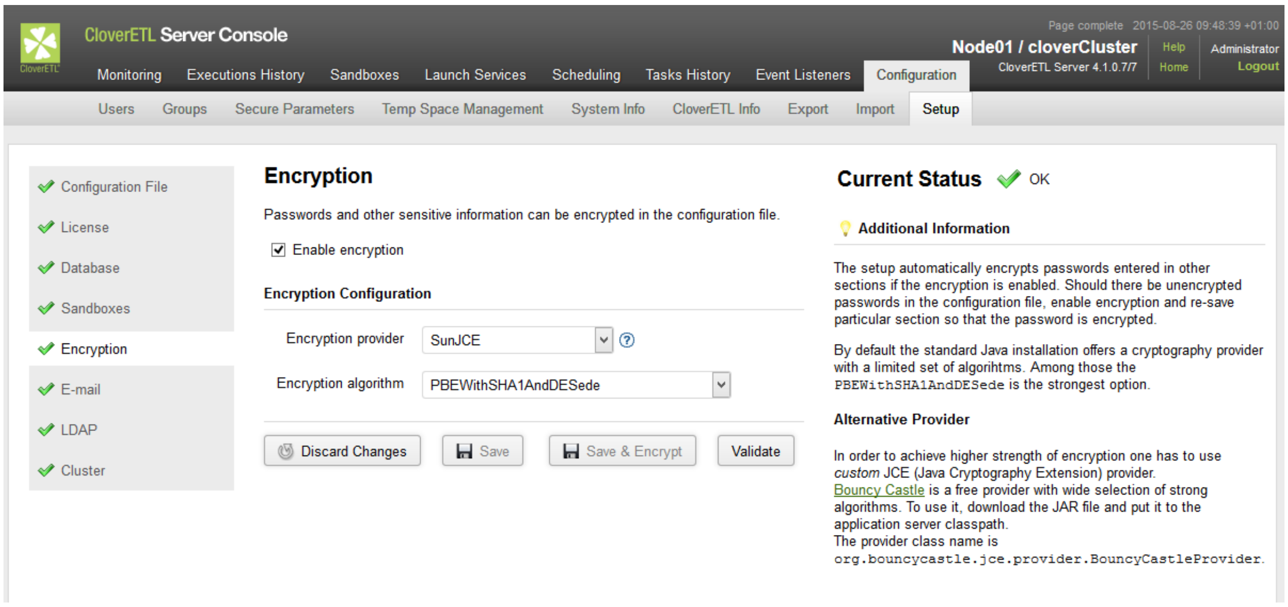
Additional Information

You can use `${variable}` notation to reference system properties and environment variables in location paths.

- `${user.data.home}` - user home directory as recommended by CloverETL Server (useful when running as a Windows service)
- `${user.home}` - user home directory as detected by Java platform
- `${user.dir}` - working directory of application server process

Encryption

The **Encryption** tab lets you enable encryption of sensitive items of the configuration file. You can choose an encryption provider and an encryption algorithm. An alternative encryption provider can be used; the libs have to be added to `classpath`. (In the same way as database libraries.)



Encryption

Passwords and other sensitive information can be encrypted in the configuration file.

[✓] Enable encryption

Encryption Configuration

Encryption provider	SunJCE
Encryption algorithm	PBEWithSHA1AndDESede

Current Status OK

Additional Information

The setup automatically encrypts passwords entered in other sections if the encryption is enabled. Should there be unencrypted passwords in the configuration file, enable encryption and re-save particular section so that the password is encrypted.

By default the standard Java installation offers a cryptography provider with a limited set of algorihtms. Among those the `PBEWithSHA1AndDESede` is the strongest option.

Alternative Provider

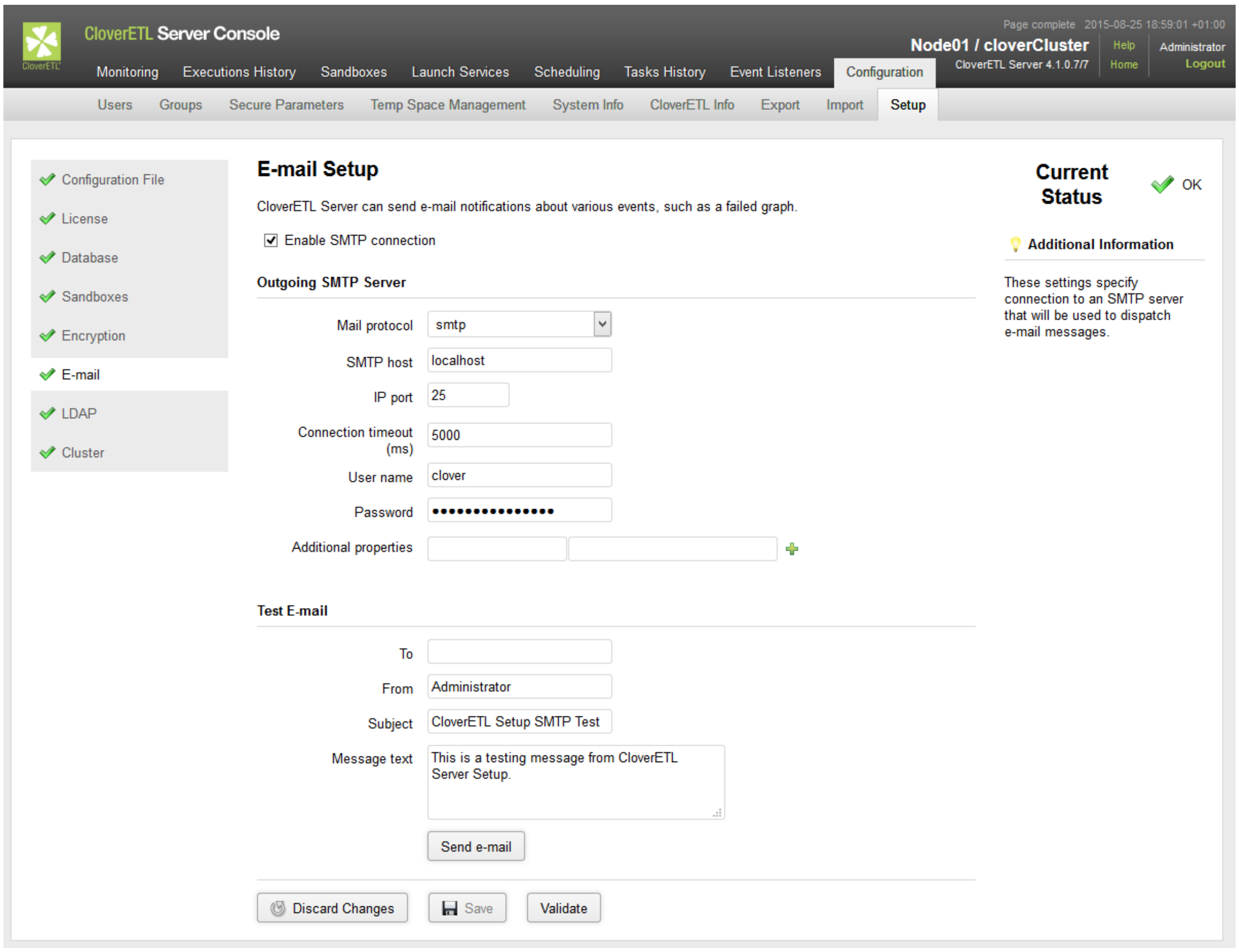
In order to achieve higher strength of encryption one has to use *custom* JCE (Java Cryptography Extension) provider. Bouncy Castle is a free provider with wide selection of strong algorithms. To use it, download the JAR file and put it to the application server classpath.
The provider class name is `org.bouncycastle.jce.provider.BouncyCastleProvider`.

The **Save & Encrypt** button saves the configuration and encrypts the passwords.

E-Mail

The **E-mail** tab lets you configure a connection to an SMTP server. The connection is necessary for reporting events on the server via e-mails.

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



E-mail Setup

CloverETL Server can send e-mail notifications about various events, such as a failed graph.

☑ Enable SMTP connection

Outgoing SMTP Server

Mail protocol	smtp
SMTP host	localhost
IP port	25
Connection timeout (ms)	5000
User name	clover
Password	•••••••••••••••
Additional properties	

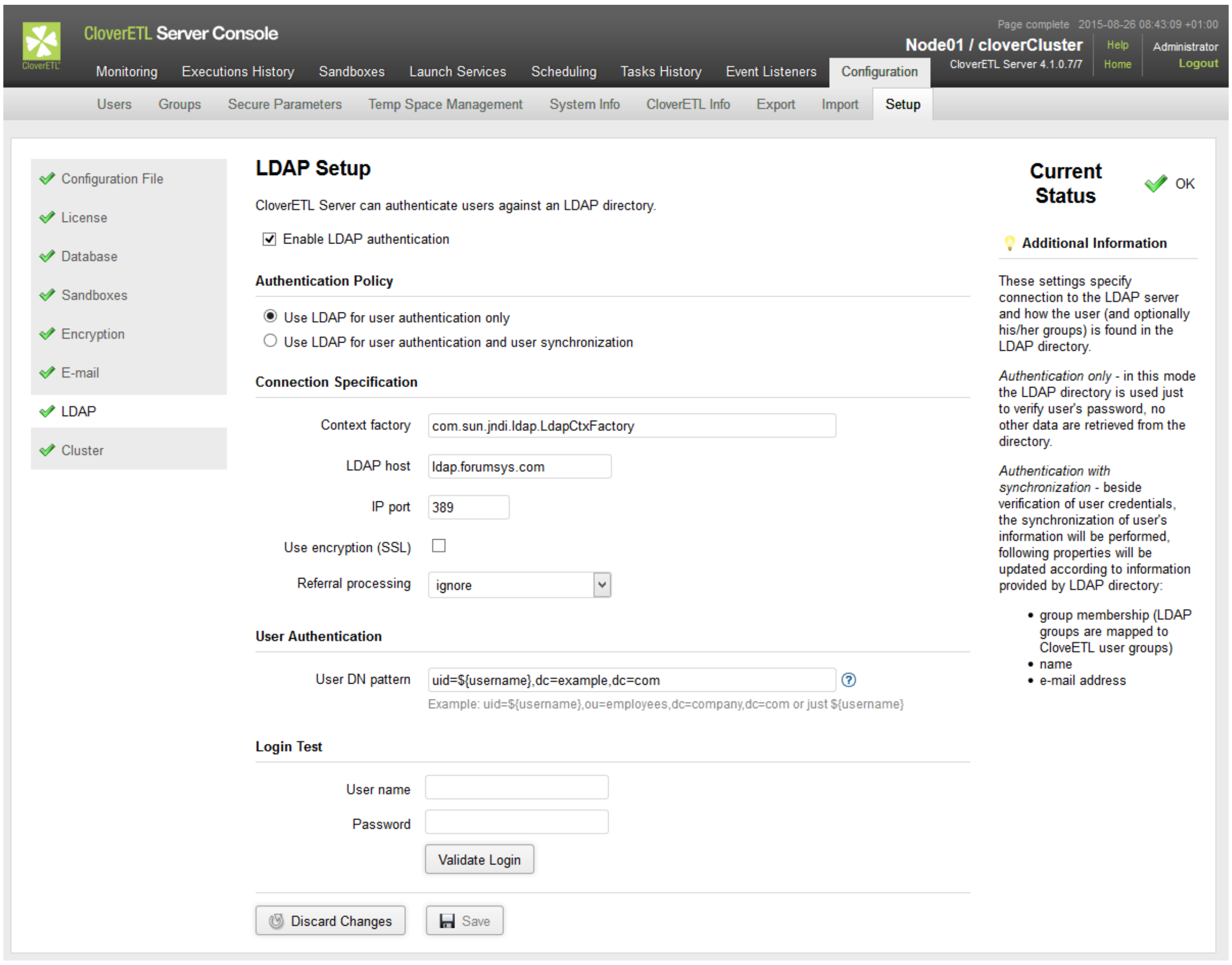
Test E-mail

To	
From	Administrator
Subject	CloverETL Setup SMTP Test
Message text	This is a testing message from CloverETL Server Setup.

These settings specify connection to an SMTP server that will be used to dispatch e-mail messages.

LDAP

The **LDAP** tab lets you use an existing LDAP database for user authentication.



LDAP Setup

CloverETL Server can authenticate users against an LDAP directory.

☑ Enable LDAP authentication

Authentication Policy

● Use LDAP for user authentication only
○ Use LDAP for user authentication and user synchronization

Connection Specification

Context factory	com.sun.jndi.ldap.LdapCtxFactory
LDAP host	ldap.forumsys.com
IP port	389
Use encryption (SSL)	☐
Referral processing	ignore

User Authentication

User DN pattern	uid=${username},dc=example,dc=com

Example: uid=${username},ou=employees,dc=company,dc=com or just ${username}

Login Test

User name	
Password	

Current Status ✓ OK

Additional Information

These settings specify connection to the LDAP server and how the user (and optionally his/her groups) is found in the LDAP directory.

Authentication only - in this mode the LDAP directory is used just to verify user's password, no other data are retrieved from the directory.

Authentication with synchronization - beside verification of user credentials, the synchronization of user's information will be performed, following properties will be updated according to information provided by LDAP directory:

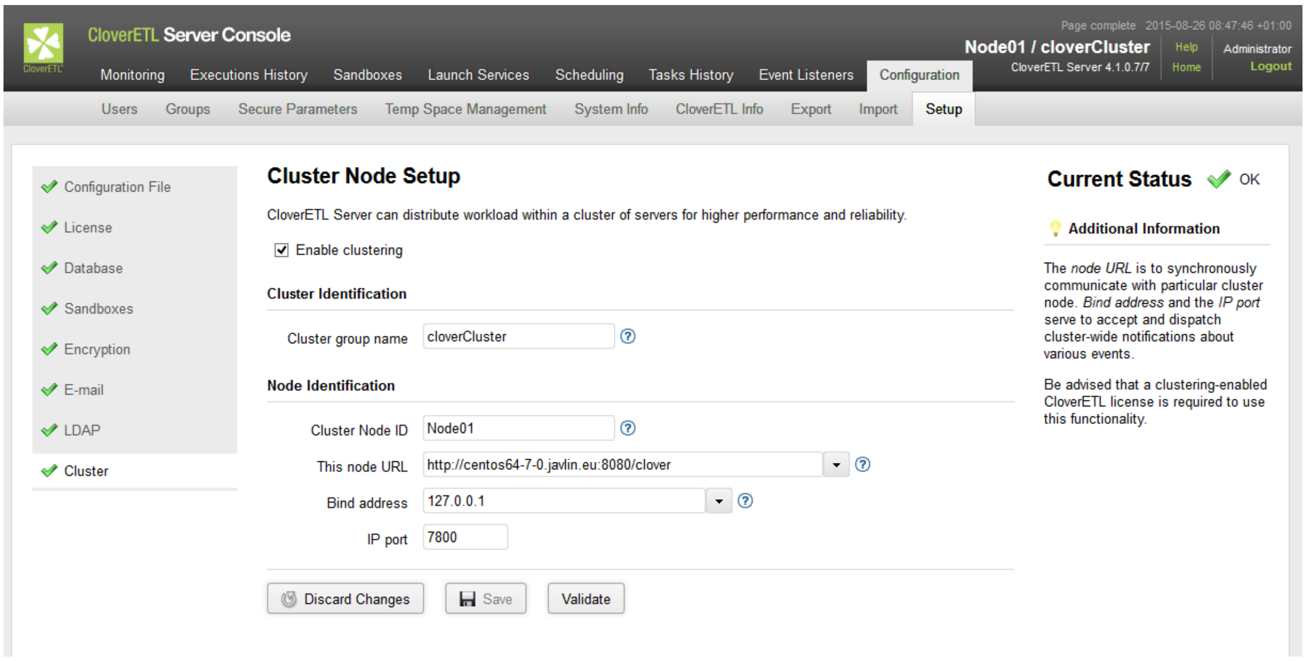
- group membership (LDAP groups are mapped to CloveETL user groups)
- name
- e-mail address

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. 111).

Cluster

The **Cluster** tab lets you configure clustering features.



Cluster Node Setup

CloverETL Server can distribute workload within a cluster of servers for higher performance and reliability.

☑ Enable clustering

Cluster Identification

Cluster group name	cloverCluster

Node Identification

Cluster Node ID	Node01
This node URL	http://centos64-7-0.javlin.eu:8080/clover
Bind address	127.0.0.1
IP port	7800

Current Status ✓ OK

Additional Information

The *node URL* is to synchronously communicate with particular cluster node. *Bind address* and the *IP port* serve to accept and dispatch cluster-wide notifications about various events.

Be advised that a clustering-enabled CloverETL license is required to use this functionality.

Note

You can use the **Setup** in a fresh installation of CloverETL Server that has not been activated yet: log into the Server Console and use the **Close** button to access the menu.

To access the **Setup** section, you need a **Server Setup** permission. See Server Setup permission (p. 128).

Chapter 8. Examples of DB Connection Configuration

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

In a clustered deployment, at least one node in the cluster must have a DB connection configured. Other nodes may have their own direct connection (to the same DB), or may use another node as a proxy for persistent operations; however, the scheduler is active only on nodes with a direct connection. See Part VI, “Cluster” (p. 246) for details about this feature, this section describes only a direct DB connection configuration.

DB Configurations and their changes may be as follows:

- Embedded Apache Derby (p. 72)
- MySQL (p. 73)
- DB2 (p. 74)
- Oracle (p. 77)
- Microsoft SQL Server (p. 78)
- Postgre SQL (p. 80)
- JNDI DB DataSource (p. 81)

See Database servers (p. 6) for officially supported versions of particular databases.

Embedded Apache Derby

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

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

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

If you use a 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. The `databases/cloverDb` part 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). You may change the default value `databases/cloverDb`.

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

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

MySQL

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

Creating database

The following steps will create database `clover_db` and user `clover` with password `clover`.

1. Create database `clover_db`, set charset and collate.

```
CREATE SCHEMA clover_db CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

2. Use clover_db as the current database.

```
USE clover_db;
```

3. Create a new user with password and host.

```
CREATE USER 'clover'@'%' IDENTIFIED BY 'clover';
```

4. Add all privileges to user 'clover' in DB clover_db.

```
GRANT ALL ON clover_db.* TO 'clover'@'%';
```

5. Reload privileges.

```
FLUSH privileges;
```

CloverETL setup

If you use a 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 a JDBC 4 compliant driver on the classpath. A JDBC Driver which doesn't meet JDBC 4 won't work properly.

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

DB2

Creating database

1. Create a dedicated user for the Clover database and set a password (UNIX/Linux).

```
useradd clover
```

```
passwd clover
```

2. Create a new database.

```
db2 "CREATE DATABASE cloverdb PAGESIZE 32768 RESTRICTIVE"
```

3. Activate the database.

```
db2 activate db cloverdb
```

4. Connect to the database.

```
db2 connect to cloverdb
```

5. Grant the user DBADM authority (DBADM authority is an administrative authority for a specific database. The database administrator possesses the privileges that are required to create objects and issue database commands. By default, DATAACCESS and ACCESSCTRL authority are also granted).

```
db2 "GRANT DBADM ON DATABASE TO USER clover"
```

6. Disconnect from database

```
db2 connect reset
```

DB2 on Linux/Windows

If you use a 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 a JDBC 4 compliant driver on the classpath. A JDBC driver which doesn't meet JDBC 4 specifications won't work properly.

Possible problems

Wrong pagesize

The *clover* database 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 the `PAGESIZE` value is not set properly, there should be an 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 a database with proper page size using the `PAGESIZE` command, e.g.:

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

The table is in the reorg pending state

On rare occasions, ALTER TABLE commands, may cause tables to remain in "reorg pending state". This behavior is specific for DB2. The ALTER TABLE DDL commands are executed only during the first start of new CloverETL Server version.

The issue may return the following error messages:

```

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

```

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

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

To solve this issue, go to DB2 console and execute the following 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 the 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 a 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 the 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.

Use `jt400ntv.jar` JDBC driver found in `/QIBM/ProdData/Java400` on the server.

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

Note

➡**Continue with:** Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

Oracle

Creating database

Run the following script to create a role, a user and a tablespace for CloverETL server.

Role: cloverRole

User: cloverUser

Password: cloverPassword

```
-- Create a new  role and grant it privileges
CREATE ROLE cloverRole NOT IDENTIFIED;

GRANT CREATE SESSION TO cloverRole;
GRANT ALTER SESSION TO cloverRole;
GRANT CREATE TABLE TO cloverRole;
GRANT CREATE SEQUENCE TO cloverRole;
GRANT CREATE TRIGGER TO cloverRole;

-- Create a new database user with password
CREATE USER cloverUser IDENTIFIED BY cloverPassword;

-- Set quota on tablespace
GRANT UNLIMITED TABLESPACE TO cloverUser;

-- Connect a new role to a new user
GRANT cloverRole TO cloverUser;
```

CloverETL setup

If you use a 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 a JDBC 4 compliant driver on the classpath. A JDBC driver which doesn't meet the JDBC 4 specifications 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>;
```

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

Microsoft SQL Server

Creating database

It is advised to use SQL Server Authentication instead of Windows Authentication. To enable it, select the server instance in Micorosft SQL Server Management Studio, go to Properties / Security / Server authentication and select SQL Server and Windows Authentication mode. The server instance needs to be restarted.

1. Create a new database

```
CREATE DATABASE clover_db;
```

2. Enable Read Committed Snapshot Isolation on the new database

```
ALTER DATABASE clover_db SET READ_COMMITTED_SNAPSHOT ON;
```

3. Create a new login role.

```
CREATE LOGIN clover with PASSWORD = 'clover', DEFAULT_DATABASE = clover_db;
```

4. Connect to the database.

```
USE clover_db;
```

5. Create a new database user.

```
CREATE USER clover FOR LOGIN clover;
```

6. Add database role membership db_owner (Members of the db_owner fixed database role can perform all configuration and maintenance activities on the database, and can also drop the database).

```
EXEC sp_addrolemember 'db_owner','clover';
```

CloverETL setup

Using MS SQL requires configuration of database server.

- run **Micorosft SQL Server Management Studio** tool;
- create a new user under **Security**/**Logins**;
- under **Databases** create a new database (e.g. 'clover') for ClovarETL server, select the user from the previous step as owner of the database;
- under database **Options**, set the **Is Read Committed Snapshot On** option to **True**;

If you use a 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;instance=SQLSERVERINSTANCE;database=clover_db
jdbc.username=user
jdbc.password=pass
jdbc.dialect=org.hibernate.dialect.SQLServerDialect
```

Please do not forget to add a JDBC 4 compliant driver on the application server classpath. A JDBC driver that does not meet the JDBC 4 specifications will not work properly.

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

Postgre SQL

Creating database

Advanced users can create their own table space

We are going to create a database for clover to use a 'user group' role which will own the database and a user role which we will add to the user group. This user role will be then used by the server to access the database.

Database name: clover_db

UserGroup: cloveretl

User: clover

Password: clover

1. Optionally, you can create new tablespace
2. Connect as postgres (default admin) to the default db postgres and execute the following commands:

```
CREATE ROLE cloveretl NOSUPERUSER NOCREATEDB NOCREATEROLE NOINHERIT NOLOGIN;
CREATE ROLE clover NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT LOGIN ENCRYPTED PASSWORD 'clover';
GRANT cloveretl TO clover;
CREATE DATABASE clover_db;
GRANT ALL ON DATABASE clover_db TO cloveretl;
REVOKE ALL ON DATABASE clover_db FROM public;
```

To separate database into its own tablespace create a tablespace before creating the database:

and use the following command to create the db:

```
CREATE DATABASE clover_db WITH OWNER cloveretl TABLESPACE tablespace_name;
```

https://www.postgresql.org/docs/9.2/static/sql-createtablespace.html

CloverETL setup

If you use a 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 a add JDBC 4 compliant driver on the classpath. A JDBC driver which doesn't meet the JDBC 4 specifications 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.

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

JNDI DB DataSource

CloverETL Server can connect to a database using JNDI DataSource, which is configured in an application server or container. However, there are some CloverETL **parameters which must be set**, otherwise the behavior 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 the sections above. #
```

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

Example of DataSource configuration in Apache Tomcat. The following context resource configuration may be added to the `[Tomcat_home]/conf/server.xml` file to the `<Host>` element.

```
<Context path="/clover" >
  <Resource name="jdbc/clover_server" auth="Container"
    type="javax.sql.DataSource" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://127.0.0.1:3306/clover?useUnicode=true&amp;characterEncoding=utf8"
    username="root" password=""
    maxActive="20" maxIdle="10" maxWait="-1"/>
</Context>
```

Do not put the code above into `<GlobalNamingResources>` element, since the resource would not be visible by the CloverETL webapp.

Note

The resource configuration may also be added to the context file `[Tomcat_home]/conf/Catalina/localhost/clover.xml`.

Note

Special characters typed in the context file have to be specified as XML entities, e.g. ampersand "&" as "&amp;" etc.

Important

The default jndi pool **DBCP** in **Apache Tomcat** does not handle connections in efficient way. With the **DBCP** jndi pool, low performance can be seen if DBOutputTable with returning statement is used.

Use **tomcat-jdbc-pool** instead. Just add `factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"` to the definition of the jndi resource. See https://tomcat.apache.org/tomcat-7.0-doc/jdbc-pool.html

See Chapter 9, List of Properties (p. 88) for list of properties.

Note

Continue with: Encrypted JNDI (p. 82) or CloverETL Server Activation (p. 45)

Encrypted JNDI

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

Encrypted JNDI on Tomcat (p. 82)
Encrypted JNDI on Jetty 9 (9.2.6) (p. 83)
Encrypted JNDI on JBoss 6.0.0 (p. 84)
Encrypted JNDI on JBoss 7 (p. 85)
Encrypted JNDI on Glassfish 3 (3.1.2.2) (p. 86)
Encrypted JNDI on WebSphere 8.5.5.0 (p. 87)
Encrypted JNDI on WebLogic (p. 87)

Encrypted JNDI on Tomcat

You need `secure-cfg-tool` to encrypt the passwords. Use the version of `secure-cfg-tool` corresponding to the version of CloverETL Server. Usage of the tool is described in Chapter 10, Secure Configuration Properties (p. 94).

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

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

Example of encrypted JNDI connection for Postgresql

Encrypt the password:

1. `./encrypt.sh -a PBEWithSHA1AndDESede`
2. 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 the classpath of the application server (`${CATALINA_HOME}/lib`). The encrypt command requires the 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)

Note

See the Jetty documentation on [Secure Password Obfuscation](#).

Configuration of a 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 the Password

Password can be obfuscated using `org.eclipse.jetty.util.security.Password` class 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 the Password

Replace the plain text password with the Call element. Its only argument is a string starting 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>
```

Note

Password in the JMS connection can also be obfuscated.

Encrypted JNDI on JBoss 6.0.0

Note

See the JBoss documentation on [Encrypting Data Source Passwords]

(In the documentation, `client/jboss-logging-spi.jar` is used; however in newer version, the `client/jboss-logging.jar` can be used instead.)

Original datasource with an 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>
```

1. Encrypt the data source password:

   - **Unix-like systems:**

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

   - **Windows system:**

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

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

2. 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>
```

3. Replace authentication entries with a reference to the application authentication policy

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

   The 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>
```

Encrypted JNDI on JBoss 7

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

Note

See Using Encrypted DataSource Password in JBoss AS7 for details.

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>
```

1. In JBOSS_HOME directory run the 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.

2. Add a new security-domain to security-domains, the password value is a 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>
```

3. 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>
```

It is possible that the same mechanism can also be used for JMS.

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 with so called password alias:

A password alias stores a password in an 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

There are two ways to create a password alias: using create-password-alias command in a command-line admin-console utility, or in the web Server Administration Console in the Password Aliases section (Domain->Password Aliases).

Replacing the Password with the Password Alias

Replace the password (the attribute value) with a ${ALIAS=password_alias_name} string, 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 Administration Server Console mentions a lower case keyword (alias); if it doesn't work, try changing to upper case (ALIAS).

Note

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. (see How to Create a WAS JDBC Provider, J2C Authentication Alias, and Data Source for the IBM i).

The same mechanism can also be used for JMS connection (see IBM's instructions on Configuring an external JMS provider).

Encrypted JNDI on WebLogic

Password in a 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 (see Oracle's instructions on Configuring an external JMS provider).

Note

Continue with: CloverETL Server Activation (p. 45)

Chapter 9. List of Properties

Table 9.1. General configuration

key	description	default
config.file	location of a CloverETL Server configuration file	[working_dir]/ cloverServer.properties
license.file	location of a CloverETL Server licence file (license.dat)	
engine.config.file	location of a CloverETL engine configuration properties file	properties file packed with CloverETL
sandboxes.home	This property is primarily intended to be used as a 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. The sandbox path may still be specified as an absolute path, but placeholder has some significant advantages: * sandbox definition may be exported/imported to another environment with a different directory structure * user creating sandboxes doesn't have to care about physical location on the filesystem * each node in cluster environment may have a different "sandboxes.home" value, so the directory structure doesn't have to be identical The default value uses content of a "user.data.home" configuration property, 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 the 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, so their values are not visible for web GUI users. The values are replaced by a 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 an absolute path to some "source" of additional CloverETL engine plugins. These plugins are not a substitute for plugins packed in WAR. "Source" may be a directory or a zip file. Both, a directory and a zip, must contain a subdirectory for each plugin. Changes in the directory or the ZIP file apply only when the server is restarted. For details see Chapter 27, Extensibility - CloverETL Engine Plugins (p. 244).	empty
datasource.type	Set this explicitly to JNDI if you need CloverETL Server to connect to a DB using JNDI datasource. In such case, "datasource.jndiName" and "jdbc.dialect" parameters must be set properly. Possible values: JNDI | JDBC	JDBC

key	description	default
datasource.jndiName	JNDI location of a 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 the path relative to a 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 in which all new users are included. 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	List of features which are accessible using HTTP and which should be protected by Basic HTTP Authentication. The list has form of semi-colon separated items; 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	List of features which are accessible using HTTP and which should be protected by HTTP Digest Authentication. The list has form of semi-colon separated items. 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		false

key	description	default
	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. This option is not automatically enabled when any feature is specified in the security.digest_authentication.features_list property.	
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 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 the timer, running file event listener checks (in milliseconds). See File Event Listeners (remote and local) (p. 212) 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 a "clover.smtp.additional." prefix are automatically added (without the prefix) to the Properties instance passed to the Mailer. May be useful for some protocol specific parameters. The prefix is removed.	
logging.project_name	Used in log messages where it is necessary to name the product name.	CloverETL
logging.default_subdir	Name of a default subdirectory for all server logs; it is relative to the path specified by system property "java.io.tmpdir". Don't specify as an absolute path, use properties which are intended for absolute path.	cloverlogs
logging.logger.server_audit.enabled	Enables logging of operations called on ServerFacade and JDBC proxy interfaces. The name of the output file is "server-audit.log". It is stored in the same directory as other CloverETL Server log files by default. The 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. 234) for details.	${java.io.tmpdir}/ [logging. default_subdir]/ launch where

key	description	default
		${java.io.tmpdir} is system property
graph.logs_path	Location, where server should store Graph run logs. See Chapter 11, Logging (p. 98) for details.	${java.io.tmpdir}/ [logging. default_subdir]/ graph where ${java.io.tmpdir} is system property
logging.appender.jobs.pattern_layout	Pattern of the jobs' log messages	%d %-5p %-3X{runId} [%t] %m%n
logging.appender.jobs.encoding	Encoding of the jobs' log files	UTF-8
temp.default_subdir	Name of a default subdirectory for server tmp files; it is relative to the path specified by system property "java.io.tmpdir".	clovertmp
graph.pass_event_params _to_graph_in_old_style	Since 3.0. It is a switch for backwards compatibility of passing parameters to the graph executed by a graph event. In versions prior to 3.0, all params are passed to executed graph. Since 3.0, just specified parameters are passed. Please see Start a Graph (p. 170) for details.	false
threadManager.pool.corePoolSize	Number of threads which are always active (running or idling). Related to a thread pool for processing server events.	4
threadManager.pool.queueCapacity	Max size of the queue (FIFO) which contains tasks waiting for an available thread. Related to a thread pool for processing server events. For queueCapacity=0, there are no waiting tasks, each task is immediately executed in an available thread or in a new thread.	0
threadManager.pool.maxPoolSize	Max number of active threads. If no thread from a core pool is available, the 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 idling threads timeout. If true, the "corePoolSize" is ignored so all idling threads may be time-outed	false
threadManager.pool.keepAliveSeconds	timeout for idling 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

key	description	default
task.archivator. archive_file_prefix	Prefix of archive files created by the archivator.	cloverArchive_
license.context_names	A comma-separated list of web-app contexts which may contain license. Each of them has to start with a slash! Works only on Apache Tomcat.	/clover-license,/ clover_license
properties_resolver.resolve_server_props.server_props_list_additional	A list of properties from a subset of properties, which values are resolved. The properties' values may use system properties or environment variables as placeholders. The values are resolved during the server startup. If the system property is changed later, the resolved CloverETL Server property value doesn't change. Users may use this property, if some property they need to resolve is missing in the property: properties_resolver.resolve_server_props.server_props_list_default. If the property to resolve is already specified by the property properties_resolver.resolve_server_props.server_props_list_default, don't add it to this property.	
properties_resolver.resolve_server_props.server_props_list_default	A list of properties from a subset of properties, which values are resolved. The properties' values may use system properties or environment variables as placeholders. Values are resolved during the server startup. If the system property is changed later, the resolved CloverETL Server property value doesn't change. Users are discouraged from modification of the property, unless it's necessary. Instead, users may add more properties by modifying property: properties_resolver.resolve_server_props.server_props_list_additional	sandboxes.home, sandboxes.home.local, sandboxes.home.partitioned, cluster.jgroups.bind_address, cluster.jgroups.start_port, cluster.jgroups.external_address, cluster.jgroups.external_port, cluster.jgroups.tcpping.initial_hosts, cluster.group.name, cluster.http.url
properties_resolver.placeholders.server_props_list_default	A list of properties from a subset of properties, that may be used as placeholders and shall be resolved if used in paths. The properties can be used if you define a path to the root of a sandbox, or to locations of local or partitioned sandboxes, or path to a script, or path in archiver job. Users are strongly discouraged from modification of the property. The property name changed since CloverETL 4.2, however the obsolete name is also still accepted to maintain backwards compatibility.	sandboxes.home, sandboxes.home.local, sandboxes.home.partitioned, user.data.home
webDav.method.propfind.maxDepth	Maximum depth for webDAV method PROPFIND. When the depth is not specified, the default is supposed to be infinite (according to the rfc2518), however it's necessary to set some limit, otherwise the webDav client might overload the server filesystem.	40

key	description	default
	Also if the depth value specified by webDAV client in the request is higher than the pre-configured max depth, only the pre-configured maximum is used.	
jvm.implementation.check.enabled	Displays warnings when unsupported Java implementation is used.	true

Table 9.2. Defaults for job execution configuration - see Job Config Properties (p. 137) for details

key	description	default
executor.tracking_interval	An interval in milliseconds for scanning of a current status of a running graph. The shorter interval, the bigger log file.	2000
executor.log_level	Log level of graph runs. TRACE | DEBUG | INFO | WARN | ERROR	INFO
executor.max_job_tree_depth	Defines maximal depth of the job execution tree, e.g. for recursive job it defines the 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 can a graph instance 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 a graph run classpath.	
executor.skip_check_config	Disables check of graph configuration. Increases performance of a graph execution; however, it may be useful during graph development.	true
executor.password	This property is deprecated. The password for decoding encoded DB connection passwords.	
executor.verbose_mode	If true, more descriptive logs of graph runs are generated.	true
executor.use_jmx	If true, the graph executor registers jmx mBean of the running graph.	true
executor.debug_mode	If true, edges with enabled debug store data into files in debug directory.	false

See Chapter 29, Clustering Features (p. 247) for more properties.

Chapter 10. Secure Configuration Properties

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

Basic Utility Usage (p. 94)
Advanced Usage - Custom Settings (p. 95)

Basic Utility Usage

1. Get the 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 the script given for your operating system, `encrypt.bat` for MS Windows, `encrypt.sh` for Linux. You will be asked for inserting a value of a 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 to configure the way the values are encrypted, see Advanced Usage - Custom Settings (p. 95)

3. Encrypted string has *conf#encrypted_property* format. The encrypted string can be used as a value of a configuration property in the properties file, `clover.xml` file or `web.xml` file (see details about configuration sources in Chapter 6, Configuration Sources and Their Priorities (p. 61)).

   Example (snippet of a 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, **java -jar secure-cfg-tool.jar** command can be used.

Important

Values encrypted by a Secure parameter form (Chapter 13, Secure Parameters (p. 107)) **cannot** be used as a value of a configuration property.

Advanced Usage - Custom Settings

The way of encrypting configuration values described above 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 a custom provider class (it will be added to the 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

A configuration file is a 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. 60)). 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 (p. 98)
Another Useful Logging Settings (p. 98)
Access Log in Apache Tomcat (p. 99)
Graph Run Logs (p. 99)
Server Audit Logs (p. 99)

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 a 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 a 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 with the CloverETL "graph.logs_path" property. This property does not influence main Server logs.

Server Audit Logs

Server Audit Log logs operations called on *ServerFacade* and *JDBC proxy* interfaces.

Audit logging can be enabled by setting (adding) the value of CloverETL property `logging.logger.server_audit.enabled` to true. In server GUI, you can change the property value in **Configuration** →**Setup** →**Configuration File**. *Audit logging* is disabled by default.

The name of output file is `server-audit.log`. The file is 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 exiting 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.

To access this administration section, you need Temp Space Management permission (p. 128).

Overview

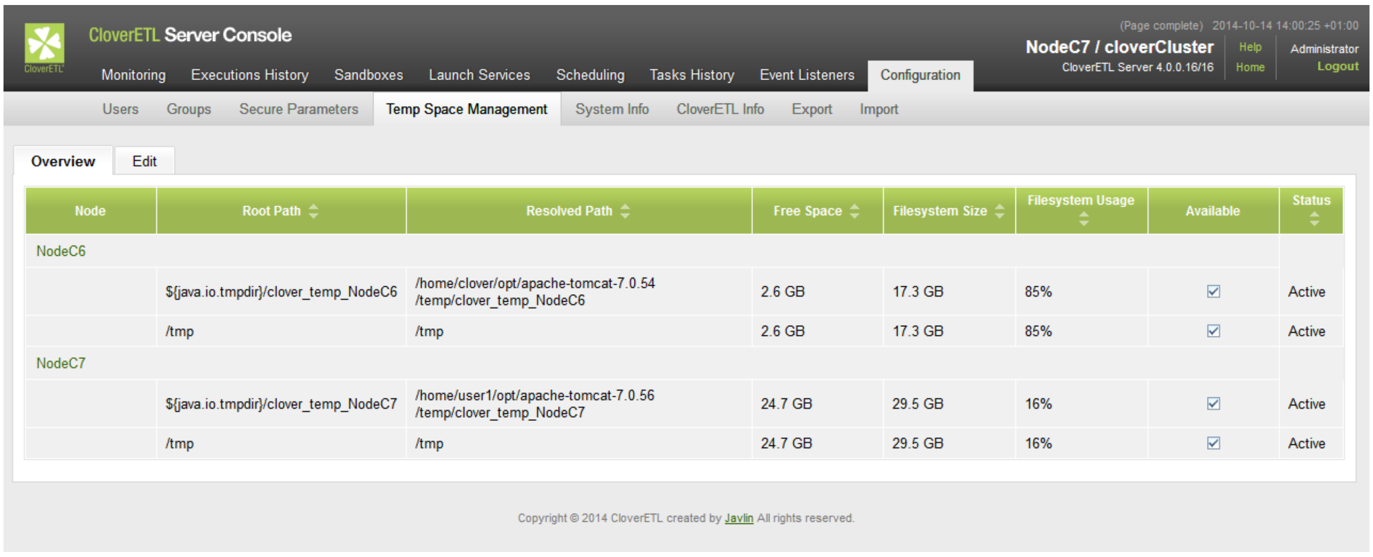
The overview of temp spaces defined in CloverETL Server is available under *Configuration > Temp space management > Overview*

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. 103).



Node	Root Path	Resolved Path	Free Space	Filesystem Size	Filesystem Usage	Available	Status
NodeC6							
	${java.io.tmpdir}/clover_temp_NodeC6	/home/clover/opt/apache-tomcat-7.0.54 /temp/clover_temp_NodeC6	2.6 GB	17.3 GB	85%	✓	Active
	/tmp	/tmp	2.6 GB	17.3 GB	85%	✓	Active
NodeC7							
	${java.io.tmpdir}/clover_temp_NodeC7	/home/user1/opt/apache-tomcat-7.0.56 /temp/clover_temp_NodeC7	24.7 GB	29.5 GB	16%	✓	Active
	/tmp	/tmp	24.7 GB	29.5 GB	16%	✓	Active

Copyright © 2014 CloverETL created by Javlin All rights reserved.

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 (p. 102)
Adding Temp Space (p. 102)
Using environment variables and system properties (p. 103)
Disabling Temp Space (p. 104)
Enabling Temp Space (p. 105)
Removing Temp Space (p. 105)

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:

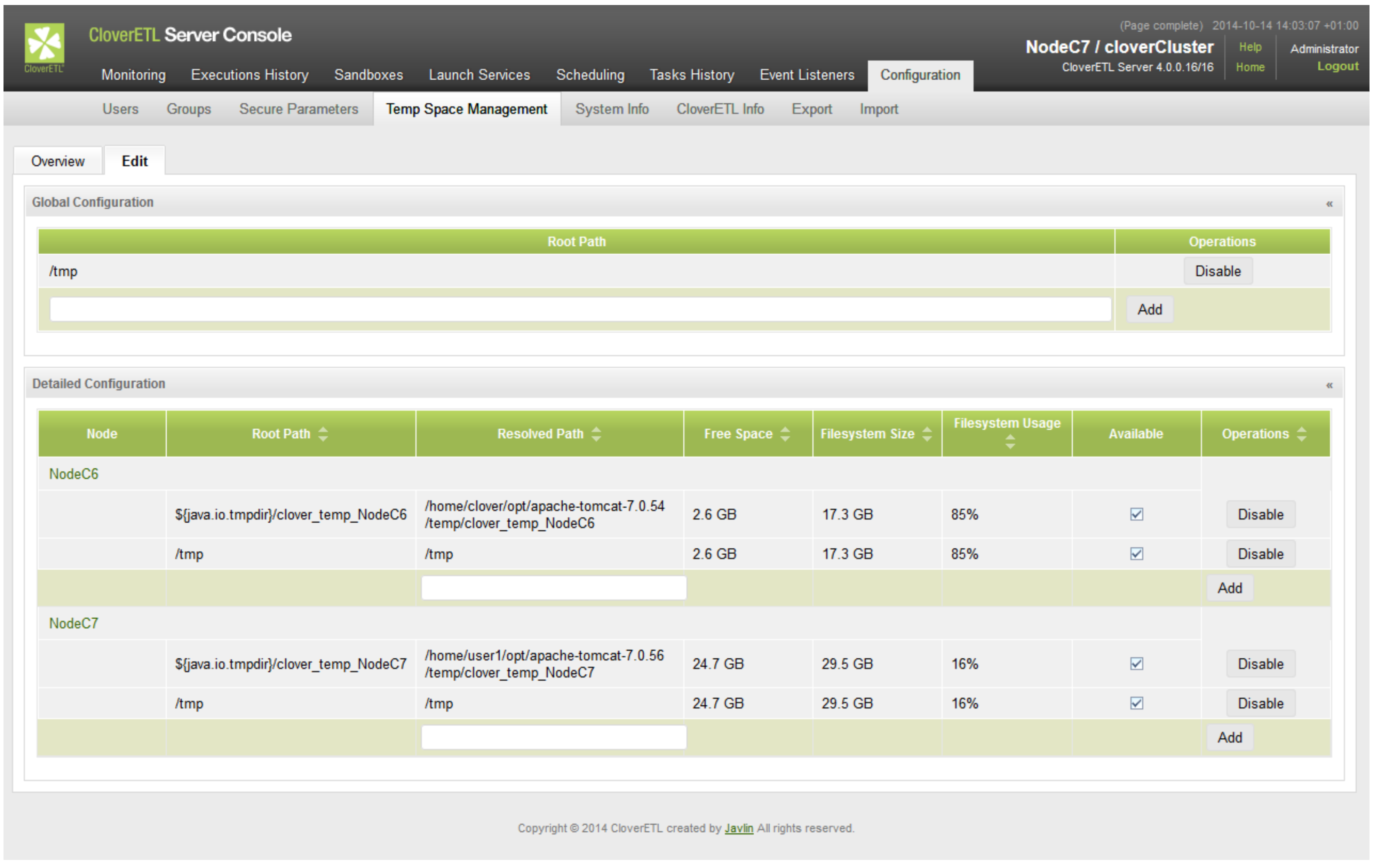
- `${java.io.tmpdir}/clover_temp` in case of a standalone server
- `${java.io.tmpdir}/clover_temp_<node_id>` in case of server cluster

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.



Global Configuration

Root Path	Operations
/tmp	Disable
	Add

Detailed Configuration

Node	Root Path	Resolved Path	Free Space	Filesystem Size	Filesystem Usage	Available	Operations
NodeC6							
	${java.io.tmpdir}/clover_temp_NodeC6	/home/clover/opt/apache-tomcat-7.0.54 /temp/clover_temp_NodeC6	2.6 GB	17.3 GB	85%	✓	Disable
	/tmp	/tmp	2.6 GB	17.3 GB	85%	✓	Disable
							Add
NodeC7							
	${java.io.tmpdir}/clover_temp_NodeC7	/home/user1/opt/apache-tomcat-7.0.56 /temp/clover_temp_NodeC7	24.7 GB	29.5 GB	16%	✓	Disable
	/tmp	/tmp	24.7 GB	29.5 GB	16%	✓	Disable
							Add

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.

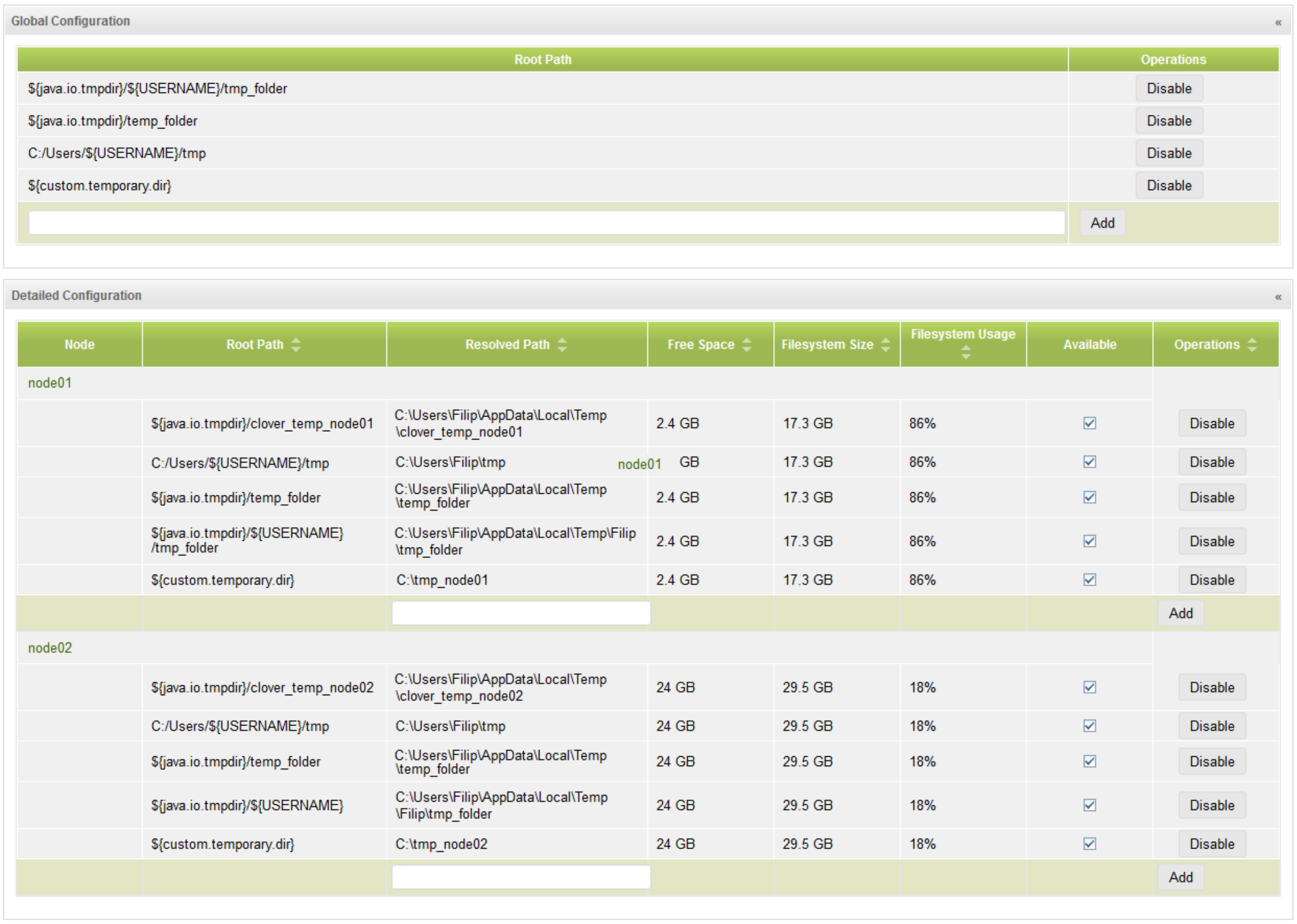
Examples:

- 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 `-Dcustom.temporary.dir=C:\tmp_node01` parameter. Node `node02` has been started with `-Dcustom.temporary.dir=C:\tmp_node02` parameter. 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`.



Global Configuration

Root Path	Operations
${java.io.tmpdir}/${USERNAME}/tmp_folder	Disable
${java.io.tmpdir}/temp_folder	Disable
C:/Users/${USERNAME}/tmp	Disable
${custom.temporary.dir}	Disable
	Add

Detailed Configuration

Node	Root Path	Resolved Path	Free Space	Filesystem Size	Filesystem Usage	Available	Operations
node01							
	${java.io.tmpdir}/clover_temp_node01	C:\Users\Filip\AppData\Local\Temp \clover_temp_node01	2.4 GB	17.3 GB	86%	✓	Disable
	C:/Users/${USERNAME}/tmp	C:\Users\Filip\tmp	node01 GB	17.3 GB	86%	✓	Disable
	${java.io.tmpdir}/temp_folder	C:\Users\Filip\AppData\Local\Temp \temp_folder	2.4 GB	17.3 GB	86%	✓	Disable
	${java.io.tmpdir}/${USERNAME} /tmp_folder	C:\Users\Filip\AppData\Local\Temp\Filip \tmp_folder	2.4 GB	17.3 GB	86%	✓	Disable
	${custom.temporary.dir}	C:\tmp_node01	2.4 GB	17.3 GB	86%	✓	Disable
							Add
node02							
	${java.io.tmpdir}/clover_temp_node02	C:\Users\Filip\AppData\Local\Temp \clover_temp_node02	24 GB	29.5 GB	18%	✓	Disable
	C:/Users/${USERNAME}/tmp	C:\Users\Filip\tmp	24 GB	29.5 GB	18%	✓	Disable
	${java.io.tmpdir}/temp_folder	C:\Users\Filip\AppData\Local\Temp \temp_folder	24 GB	29.5 GB	18%	✓	Disable
	${java.io.tmpdir}/${USERNAME}	C:\Users\Filip\AppData\Local\Temp \Filip\tmp_folder	24 GB	29.5 GB	18%	✓	Disable
	${custom.temporary.dir}	C:\tmp_node02	24 GB	29.5 GB	18%	✓	Disable
							Add

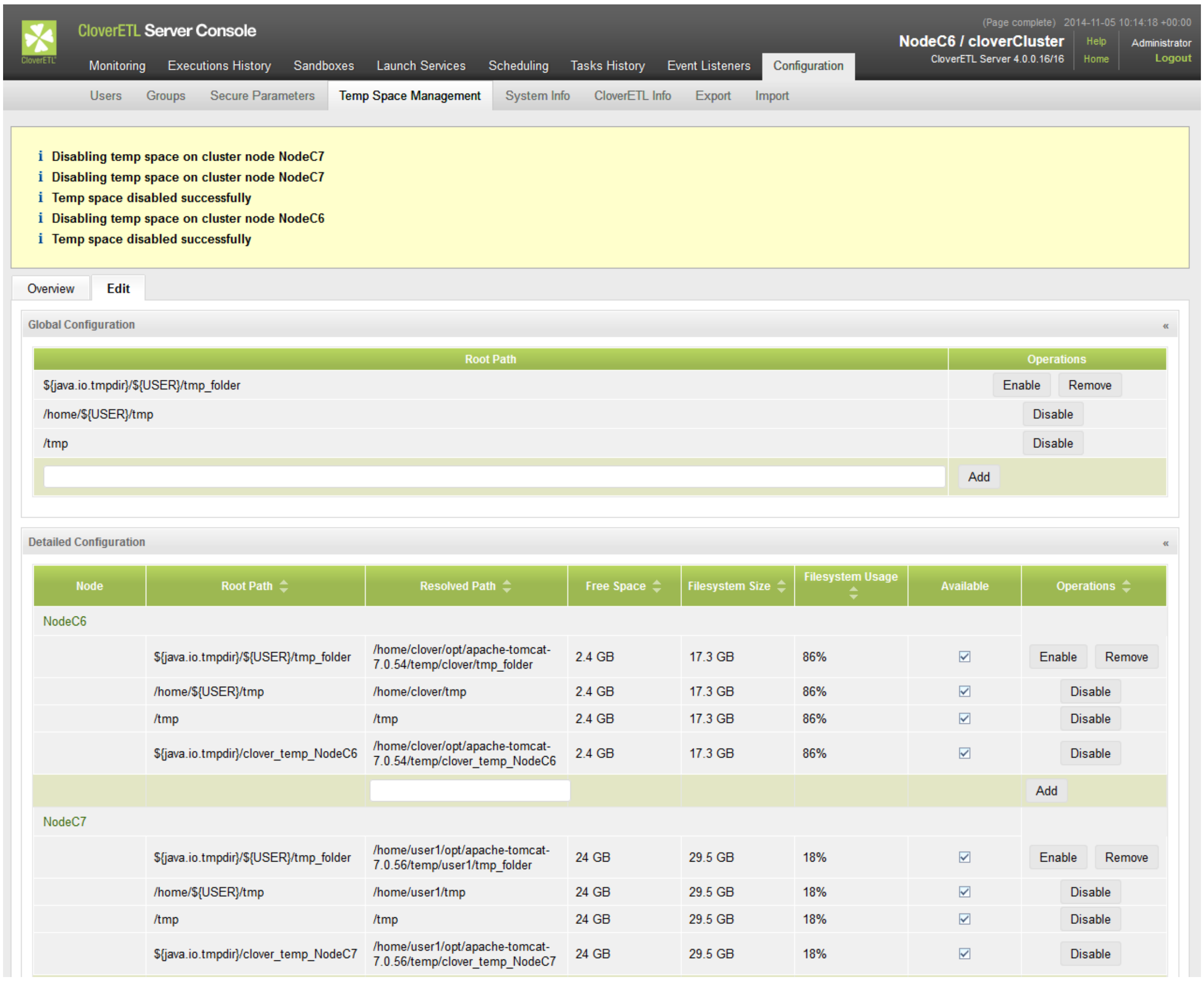
Figure 12.3. Temp spaces using environment variables and system properties

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.



i Disabling temp space on cluster node NodeC7
i Disabling temp space on cluster node NodeC7
i Temp space disabled successfully
i Disabling temp space on cluster node NodeC6
i Temp space disabled successfully

Global Configuration

Root Path	Operations
${java.io.tmpdir}/${USER}/tmp_folder	Enable Remove
/home/${USER}/tmp	Disable
/tmp	Disable
	Add

Detailed Configuration

Node	Root Path	Resolved Path	Free Space	Filesystem Size	Filesystem Usage	Available	Operations
NodeC6							
	${java.io.tmpdir}/${USER}/tmp_folder	/home/clover/opt/apache-tomcat-7.0.54/temp/clover/tmp_folder	2.4 GB	17.3 GB	86%	✓	Enable Remove
	/home/${USER}/tmp	/home/clover/tmp	2.4 GB	17.3 GB	86%	✓	Disable
	/tmp	/tmp	2.4 GB	17.3 GB	86%	✓	Disable
	${java.io.tmpdir}/clover_temp_NodeC6	/home/clover/opt/apache-tomcat-7.0.54/temp/clover_temp_NodeC6	2.4 GB	17.3 GB	86%	✓	Disable
							Add
NodeC7							
	${java.io.tmpdir}/${USER}/tmp_folder	/home/user1/opt/apache-tomcat-7.0.56/temp/user1/tmp_folder	24 GB	29.5 GB	18%	✓	Enable Remove
	/home/${USER}/tmp	/home/user1/tmp	24 GB	29.5 GB	18%	✓	Disable
	/tmp	/tmp	24 GB	29.5 GB	18%	✓	Disable
	${java.io.tmpdir}/clover_temp_NodeC7	/home/user1/opt/apache-tomcat-7.0.56/temp/clover_temp_NodeC7	24 GB	29.5 GB	18%	✓	Disable

Figure 12.4. Disable operation reports action performed

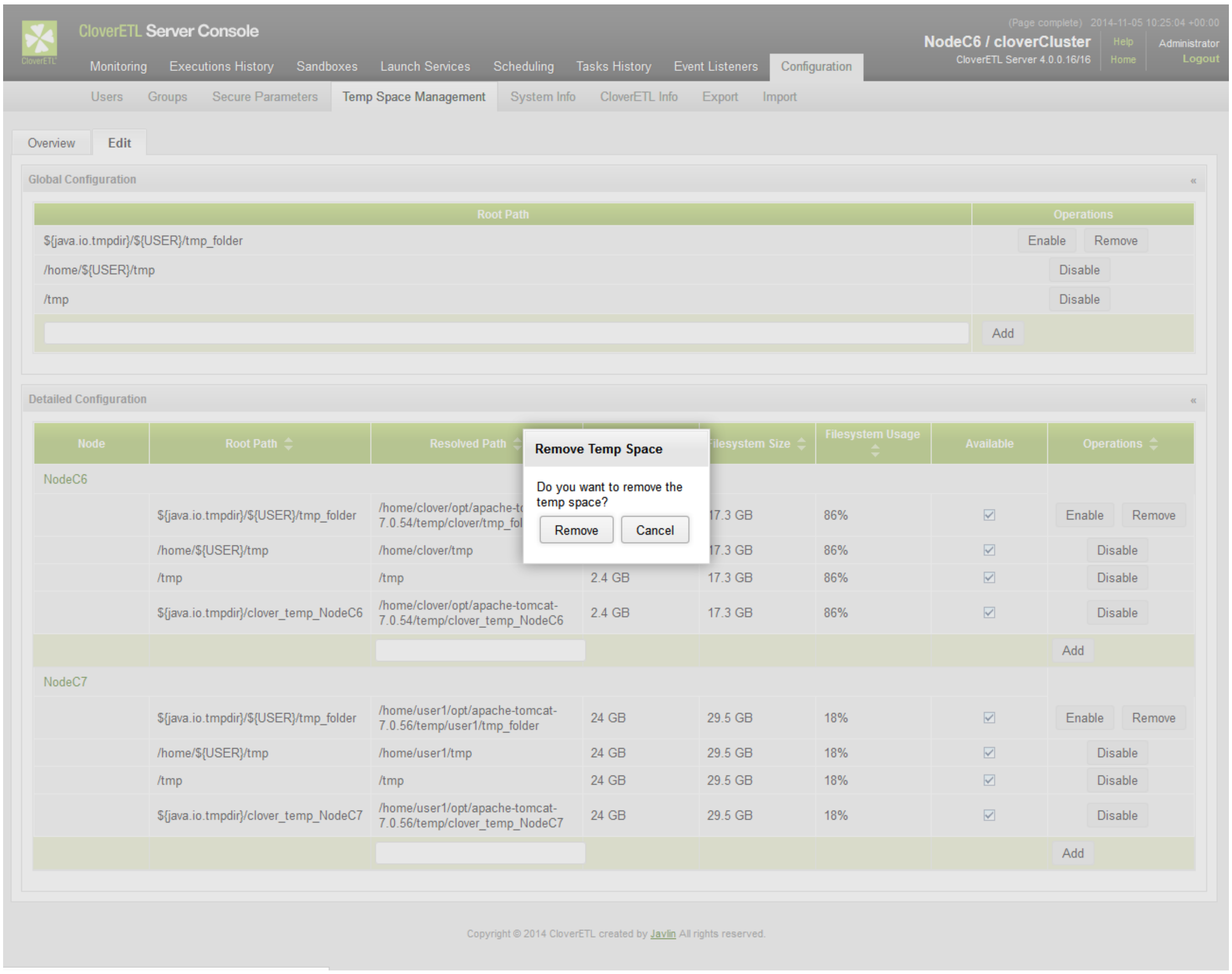
Enabling Temp Space

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 the disabled temp space may be removed. If there are any running jobs using the temp space, the 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 notification panel. The available options are:

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



CloverETL Server Console

(Page complete) 2014-11-05 10:25:04 +00:00
NodeC6 / cloverCluster
CloverETL Server 4.0.0.16/16

Global Configuration

Root Path	Operations
${java.io.tmpdir}/${USER}/tmp_folder	Enable Remove
/home/${USER}/tmp	Disable
/tmp	Disable
	Add

Detailed Configuration

Node	Root Path	Resolved Path		Filesystem Size	Filesystem Usage	Available	Operations
NodeC6							
	${java.io.tmpdir}/${USER}/tmp_folder	/home/clover/opt/apache-t 7.0.54/temp/clover/tmp_fol		17.3 GB	86%	✓	Enable Remove
	/home/${USER}/tmp	/home/clover/tmp		17.3 GB	86%	✓	Disable
	/tmp	/tmp	2.4 GB	17.3 GB	86%	✓	Disable
	${java.io.tmpdir}/clover_temp_NodeC6	/home/clover/opt/apache-tomcat-7.0.54/temp/clover_temp_NodeC6	2.4 GB	17.3 GB	86%	✓	Disable
							Add
NodeC7							
	${java.io.tmpdir}/${USER}/tmp_folder	/home/user1/opt/apache-tomcat-7.0.56/temp/user1/tmp_folder	24 GB	29.5 GB	18%	✓	Enable Remove
	/home/${USER}/tmp	/home/user1/tmp	24 GB	29.5 GB	18%	✓	Disable
	/tmp	/tmp	24 GB	29.5 GB	18%	✓	Disable
	${java.io.tmpdir}/clover_temp_NodeC7	/home/user1/opt/apache-tomcat-7.0.56/temp/clover_temp_NodeC7	24 GB	29.5 GB	18%	✓	Disable
							Add

Remove Temp Space

Do you want to remove the temp space?

Remove Cancel

Copyright © 2014 CloverETL created by Javlin All rights reserved.

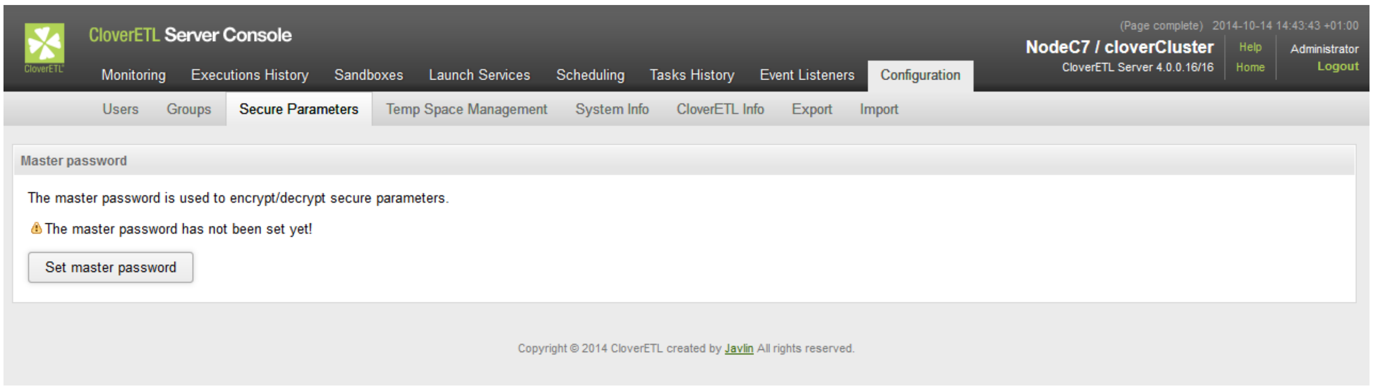
Figure 12.5. Remove operation asks for confirmation in case there are data present in the temp space

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 only available 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.

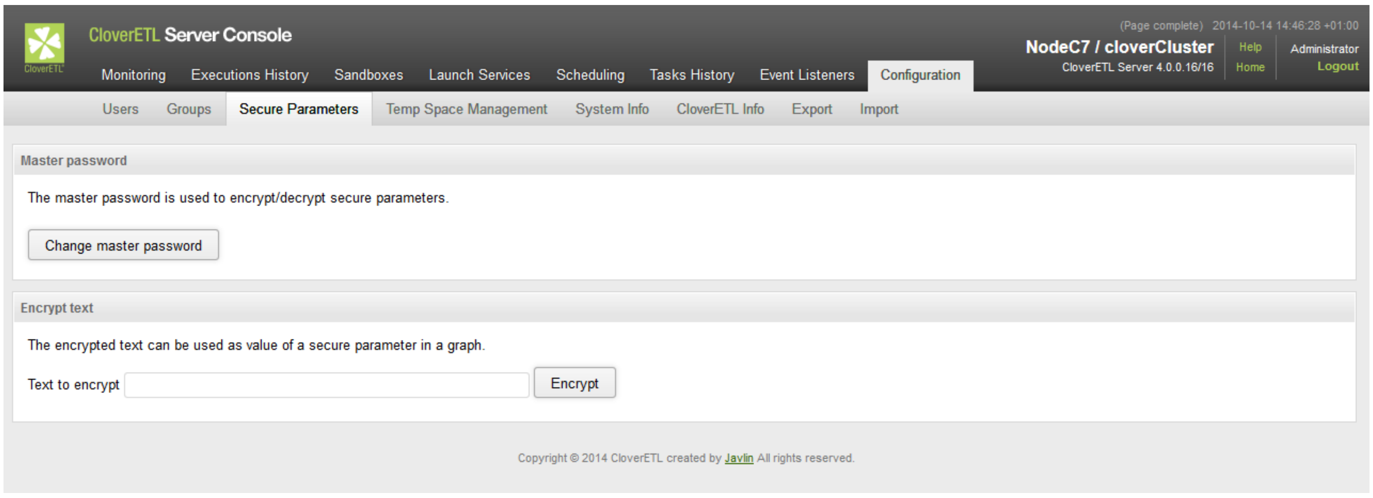


The master password is used to encrypt/decrypt secure parameters.

The master password has not been set yet!

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 decrypted 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.



The master password is used to encrypt/decrypt secure parameters.

The encrypted text can be used as value of a secure parameter in a graph.

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 decrypted 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 another server with a different master password. So it is highly recommended to use the 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 further 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.password	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 *java.security.Provider* interface. 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 *security.job_parameters.encryptor.providerClassName* attribute to *org.bouncycastle.jce.provider.BouncyCastleProvider* in file `WEB-INF/config.properties`.

4. Set value of the *security.job_parameters.encryptor.algorithm* attribute 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 the 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 a Server administrator before the the user can log in.

What does the CloverETL do to authenticate an LDAP user?

1. User specifies the LDAP credentials in a login form to the Server web console
2. CloverETL Server looks up a 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 overridden 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 an 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 the section called “LDAP” (p. 68) in Chapter 7, Setup (p. 63).

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 an attribute with list of assigned users. It's "member" attribute usually.

In case the relation is specified 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
```

Users

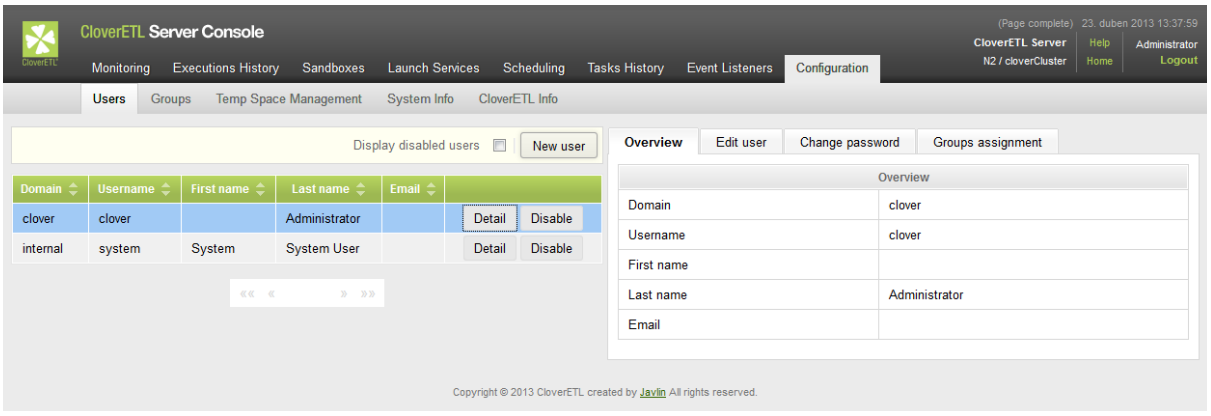
This section is intended to users management. It offers features in dependence on user's permissions. I.e., user may enter this section, but cannot modify anything. Or user may modify, but cannot create new users.

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 an 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.



Domain	Username	First name	Last name	Email	
clover	clover		Administrator		Detail Disable
internal	system	System	System User		Detail Disable

Overview	
Domain	clover
Username	clover
First name	
Last name	Administrator
Email	

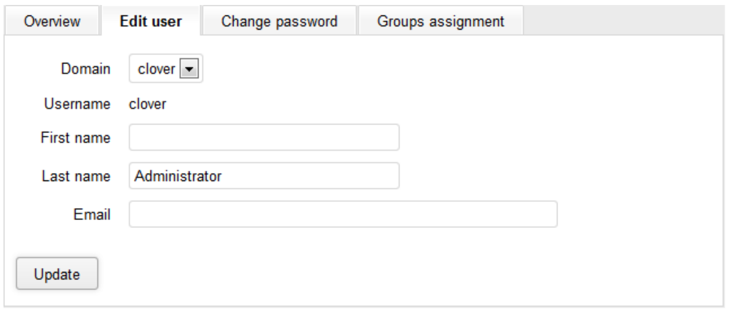
Figure 14.1. Web GUI - section "Users" under "Configuration"

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	
E-mail	E-mail address which may be used by CloverETL administrator or by CloverETL server for automatic notifications. See Send an Email (p. 163) for details.

Edit user record

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

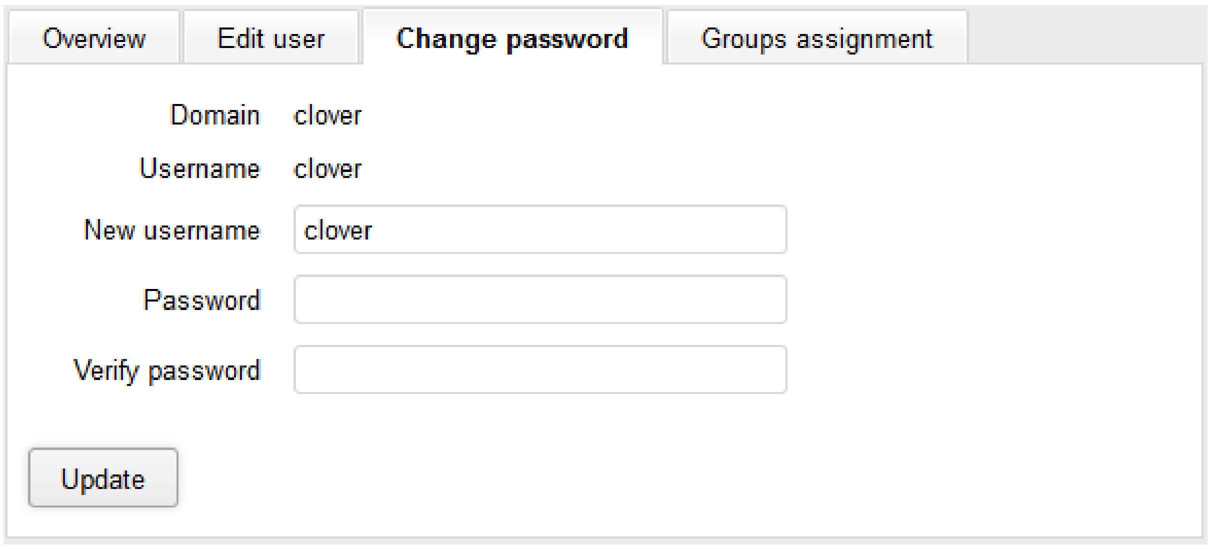


Domain	clover
Username	clover
First name	
Last name	Administrator
Email	

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.



Domain	clover
Username	clover
New username	clover
Password	
Verify password	

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 Groups (p. 117) for details about permissions.

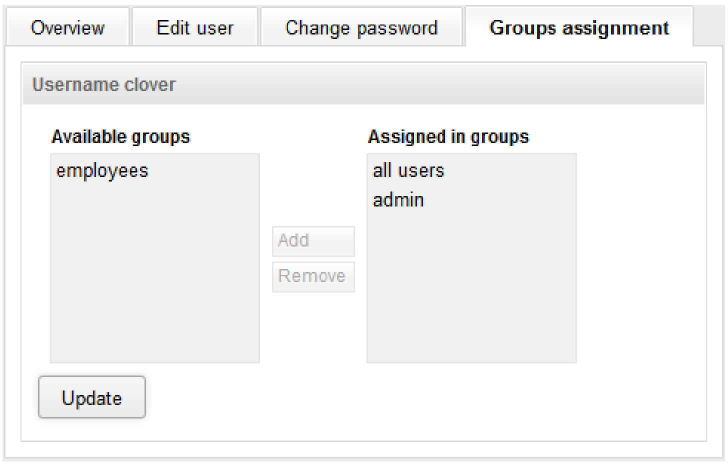


Figure 14.4. Web GUI - groups assignment

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.

Groups

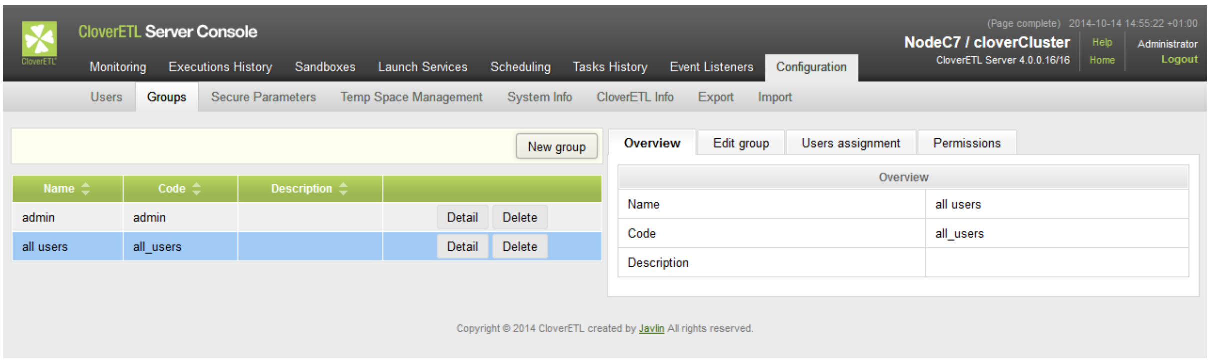
Group is an abstract set of users, which gives assigned users some permissions. So it is not necessary to specify permission for each single user.

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. 132) 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. 234) 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.



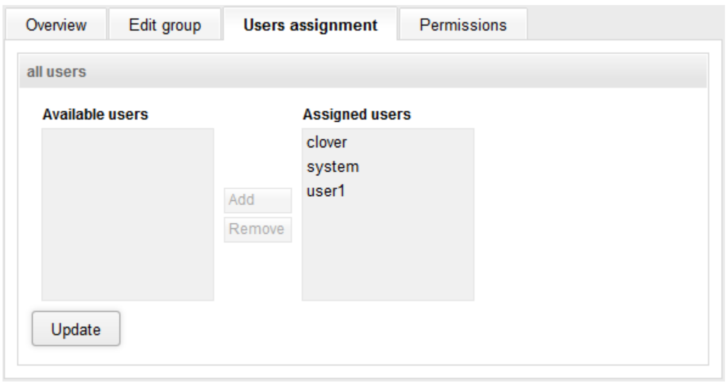
Name	Code	Description		
admin	admin		Detail	Delete
all users	all_users		Detail	Delete

Overview	
Name	all users
Code	all_users
Description	

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.



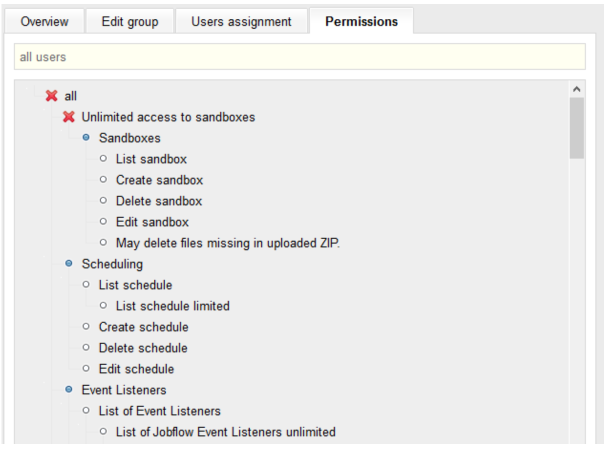
clover
system
user1

Figure 14.6. Web GUI - users assignment

Groups permissions

Groups permissions are structured as a tree, where permissions are inherited from the 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.



```
✖ all
  ✖ Unlimited access to sandboxes
    ● Sandboxes
      ○ List sandbox
      ○ Create sandbox
      ○ Delete sandbox
      ○ Edit sandbox
      ○ May delete files missing in uploaded ZIP
  ● Scheduling
    ○ List schedule
      ○ List schedule limited
    ○ Create schedule
    ○ Delete schedule
    ○ Edit schedule
  ● Event Listeners
    ○ List of Event Listeners
      ○ List of Jobflow Event Listeners unlimited
```

Figure 14.7. Tree of permissions

With no of the following privileges, user can: login to the server console, create server project (in Designer) from its own sandbox, create a file in its own existing sandbox, and run graphs.

- **all**

  A user with this permission has all available permissions. Admin group has all permissions by default.

  - **Unlimited access to sandboxes**

    This permission allows user to perform operations on all sandboxes, even if the sandbox accessibility is not specified explicitly.

    Unlimited access to sandboxes permission does not include the suspend sandbox permission (p. 125).

    - **Sandboxes**

      This permission allows user work with sandboxes. This permission contains all the permissions below. The user can perfom operations only on sandboxes owned by himself or on sandboxes with explicitly added access to him.

See Chapter 15, Server Side Job Files - Sandboxes (p. 129).

- **List sandbox**

  In server web interface, this permission allows user to list her sandboxes and list sandboxes with read permission granted to the user's group.

  In server web interface, this permission is necessary to create, edit, or delete sandboxes.

  Within a sandbox with write access granted, user can edit or remove files and create or delete directories even without this permission.

- **Create sandbox**

  This permission allows user to create a new sandbox.

  If the sandbox is to be created in web interface, the user is required to have the list sandbox permission (p. 119).

- **Delete sandbox**

  This permission allows user to delete a sandbox.

  If the sandbox is to be deleted in web interface, the user is required to have the list sandbox permission (p. 119).

- **Edit sandbox**

  This permission allows user to edit a sandbox.

  If the sandbox is to be modified in web interface, the user is required to have the list sandbox permission (p. 119).

- **May delete files missing in uploaded ZIP**

  In **Sandbox** →**Upload ZIP**, this permission allows user to use a checkbox to delete files missing in the ZIP to be uploaded. If the user does not have this permission, the checkbox to delete mission files in ZIP is not displayed.

  If the sandbox is to be uploaded from a ZIP file in server web interface, it is required to have the list sandbox permission (p. 119).

- **Scheduling**

  This permission allows user to manage schedules.

  See Chapter 22, Scheduling (p. 183).

  - **List schedule**

    This permission allows user to list all schedules.

    - **List schedule limited**

      This permission allows user to list the enabled schedules.

  - **Create schedule**

    This permission allows user to create a new schedule.

    The user needs the list schedule limited permission (p. 119) to access the scheduling section to create a new schedule.

  - **Delete schedule**

    This permission allows user to delete the schedule.

    User needs list schedule limited permission(p. 119) or list schedule permission (p. 119) to access the scheduling section to delete the schedule.

  - **Edit schedule**

    This permision allows user to edit the schedule.

    User needs list schedule limited permission(p. 119) or list schedule permission (p. 119) to access the scheduling section to edit the schedule.

- **Event listeners**

  This permission allows user to manage event listeners.

  See Chapter 24, Listeners (p. 194).

  - **List of Event Listeners**

    This permission allows user to list all event listeners.

    - **List of Jobflow Event Listeners unlimited**

      This permission allows user to list jobflow event listeners.

      See Jobflow Event Listeners (p. 202)

      - **List of Jobflow Event Listeners limited**

        This permission allows user to list jobflow event listeners of sandboxes the user can read from.

    - **List of Graph Event Listeners unlimited**

      This permission allows user to list all graph event listeners.

      See Graph Event Listeners (p. 195).

      - **List of Graph Event Listeners limited**

        This permission allows user to list graph event listeners from sandboxes the user can read from.

    - **List of File Event Listeners unlimited**

      This permission allows user to list all file event listeners.

      See File Event Listeners (remote and local) (p. 212).

      - **List of File Event Listeners limited**

        This permission allows user to list all file event listeners.

    - **List of JMS Event Listeners unlimited**

      This permission allows user to list all JMS listeners.

      See JMS Messages Listeners (p. 205).

    - **List of JMS Event Listeners limited**

      This permission allows user to list all JMS listeners.

  - **List of Universal Event Listeners unlimited**

    This permission allows user to list all universal event listeners.

    See Universal Event Listeners (p. 210).

    - **List of Universal Event Listeners limited**

      This permission allows user to list all universal event listeners.

      See Universal Event Listeners (p. 210).

- **Create Event Listener**

  This permission allows user to create event listeners.

  If the event listener is to be created in server web interface, the user needs to have permission to list the event listeners of the particular type.

  - **Create Jobflow Event Listener**

    This permission allows user to create a new Jobflow Event listener.

    If the jobflow event listener is to be created in server web interface, the user needs to have the list of jobflow event listeners limited permission (p. 120).

    See Jobflow Event Listeners (p. 202).

  - **Create Graph Event Listener**

    This permission allows user to create a graph event listener.

    If the graph gvent listener is to be created in server web interface, the user needs to have the list of graph event listeners limited permission (p. 120).

    See Graph Event Listeners (p. 195).

  - **Create File Event Listener**

    This permission allows user to create a graph event listener.

    If the file event listener is to be created in server web interface, the user needs to have the list of file event listeners limited permission (p. 120).

    See File Event Listeners (remote and local) (p. 212).

  - **Create JMS Listener**

    This permission allows user to create a JMS event listener.

    If the JMS event listener is to be created in server web interface, the user needs to have the list of JMS event listeners limited permission (p. 121).

    See JMS Messages Listeners (p. 205).

  - **Create Universal Event Listener**

This permission allows user to create a universal event listener.

If the universal event listener is to be created in server web interface, the user needs to have the list of universal event listeners limited permission (p. 121).

See Universal Event Listeners (p. 210).

- **Edit Event Listener**

  This permission allow user to edit an event listener.

  If the event listener is to be created in server web interface, the user needs to have permission to list event listener of the particular type.

  - **Edit Jobflow Event Listener**

    This permission allows user to edit a jobflow event listener.

    If the jobflow event listener is to be edited in server web interface, the user needs to have the list of jobflow event listeners limited permission (p. 120).

    See Jobflow Event Listeners (p. 202).

  - **Edit Graph Event Listener**

    This permission allows user to edit a graph event listener.

    If the graph event listener is to be edited in server web interface, the user needs to have the list of graph event listeners limited permission (p. 120).

    See Graph Event Listeners (p. 195).

  - **Edit File Event Listener**

    This permission allows user to edit a file event listener.

    If the file event listener is to be edited in server web interface, the user needs to have the list of file event listeners limited permission (p. 120).

    See File Event Listeners (remote and local) (p. 212).

  - **Edit JMS Event Listener**

    This permission allows user to edit a JMS event listener.

    If the JMS event listener is to be edited in server web interface, the user needs to have the list of JMS event listeners limited permission (p. 121).

  - **Edit Universal Event Listener**

    This permission allows user to edit a universal event listener.

    If the universal event listener is to be edited in server web interface, user needs to have permission list of universal event listeners limited permission (p. 121).

    See Universal Event Listeners (p. 210).

- **Delete Event Listener**
  This permission allows user to delete event listeners.

    - **Delete Jobflow Event Listener**

      This permisison allows user to delete a jobflow event listener.

      User needs to have the delete graph event listener permission (p. 123) to delete a jobflow event listener.

      It the jobflow event listener is to be deleted in server web interface, the user needs to have the list of jobflow event listeners limited permission (p. 120)

    - **Delete Graph Event Listener**

      This permission allows user to delete a graph event listener.

      If the graph event listener is to be deleted in server web interface, the user needs to have the list of graph event listeners limited permission (p. 120).

      See Graph Event Listeners (p. 195).

    - **Delete File Event Listener**

      This permission allows user to delete a file event listener.

      The user needs to have the delete graph event listener permission (p. 123) to delete a file event listener.

      If the file event listener is to be deleted in server web interface, the user needs to have the list of file event listeners limited permission (p. 120).

      See File Event Listeners (remote and local) (p. 212).

    - **Delete JMS Event Listener**

      This permission allows user to delete a JMS Event Listener.

      The user needs to have the delete graph event listener permission (p. 123) to delete a JMS event listener.

      If the graph event listener is to be deleted in server web interface, the user needs to have the list of JMS event listeners limited permission (p. 121).

    - **Delete Universal Event Listener**

      This permission allows user to delete a universal event listener.

      The user needs to have the delete graph event listener permission (p. 123) to delete universal event listener.

      If the universal event listener is to be deleted in server web interface, the user needs to have the list of universal event listeners limited permission (p. 121).

      See Universal Event Listeners (p. 210).

  - **Manual task Execution**

    This permission allows user to manually execute a task (send an email, execute a script, etc.) with an immediate effect.

    See Chapter 21, Manual Task Execution (p. 182).

- **Unlimited access to execution history**

This permission allows user to perform the same operations as unlimited access to execution history list permission (p. 124).

- **Unlimited access to execution history list**

  This permission allows user to view execution history of all jobs.

  - **Limited access to execution history list**

    This permission allows user to view execution history of jobs from sandboxes the user can read from. In Designer, this permission is required to be able to view **Execution log** in Designer's console and execution history in **Execution** tab.

- **Launch Services**

  This permission allows user to list, create, edit, and delete launch services.

  See Launch Services (p. 234).

  - **List Launch Services unlimited**

    This permission allows user to list all launch services.

    - **List Launch Services Limited**

      This permission allows user to list launch services from sandboxes the user can read from.

  - **Create Launch service**

    This permission allows user to create a new launch service.

    User has to have the create graph event listener permission (p. 121) to bind the launch service with a graph.

    If the launch service is to be created in server web interface, the user has to have the list launch services limited permission (p. 124) (or the list launch services unlimited permission (p. 124) to access the section with launch services.

  - **Delete Launch Service**

    This permission allows user to delete a launch service.

    User has to have delete graph event listener permission (p. 123) to delete a launch service.

    If the launch service is to be deleted in server web interface, the user needs to have the list launch services limited permission (p. 124) to access the section with launch services.

  - **Edit Launch Service**

    This permission allows user to edit a launch services.

    User has to have edit graph event listener (p. 122) to edit the launch service.

    If the launch service is to be edited in server web interface, the user needs to have the list launch services limited permission (p. 124) to choose the launch service in the server interface.

- **Tasks history**

  This permission allows user to access **Tasks history** section.

  See Chapter 20, Tasks (p. 162).

- **Monitoring**

  Monitoring permission grants user all its subpermissions.

  - **Monitoring section**

    This permission allows user to access the monitoring section.

    See Chapter 16, CloverETL Server Monitoring (p. 142).

  - **Suspend**

    This permission allows user to suspend the server, a cluster node, or a sandbox.

    The user needs to have the monitoring section permission (p. 125) to access the Monitoring section.

    - **Suspend server**

      This permission allows user to suspend or resume the server.

      The user needs to have the monitoring section permission (p. 125) to access the monitoring section.

    - **Suspend cluster node**

      This permission allows user to suspend or resume a cluster node.

      The user needs to have the monitoring section permission (p. 125) to access the monitoring section.

    - **Suspend sandbox**

      This permission allows user to suspend a sandbox. The user needs to have list sandbox permission (p. 119) to view the sandboxes to suspend them.

      See also Chapter 15, Server Side Job Files - Sandboxes (p. 129).

  - **Reset caches**

    Deprecated.

  - **Running jobs unlimited**

    If the graph is to be run from server web interface, the user needs to have the list sandbox permission (p. 119) to list the graphs.

    - **Running jobs limited**

      If the graph is to be run from server web interface, the user needs to have the list sandbox permission (p. 119) to list the graphs.

- **Configuration**

  This permission allows user to access the configuration section.

  - **Users**

    This permission allow user to access the **Users** section and configure user accounts.

    - **List user**

      This permission allows user to list users and access to the **Users** administation section (**Configuration →Users**)

    - **Change passwords**

      This permission allows user to change his password and to change password of another user.

      To see list of users, the user needs the list user permission (p. 125).

    - **Edit user**

      This permission allows user to change group assignment.

      To see the list of users, the user needs to have the list user permission (p. 125).

        - **Edit own profile and password**

          This permisison allows user to change his profile (first name, last name, email, and password).

          The user can access her profile in main web onsole view under username, in upper right corner of the page.

    - **Delete user**

      This permission allows user to disable a user.

      The user needs to have the list user permission (p. 125) to list available users.

    - **Create user**

      This permission allows user to create a new user.

      If the user is to be created in server web interface, the creating user needs to have the list user permission (p. 125) to list users to access this option.

    - **Groups assignement**

      This permission allows user to assign users to groups.

      The user needs to have the edit user permission (p. 126) to sucessfully finish the assigment of users to groups.

      If the user is to be created in server web interface, the creating user needs to have the list user permission (p. 125) to list users to access this option.

- **Groups**

  This permission allows user to manage groups: user can list groups, create groups, delete groups, edit the group, assign users to the group, and change permissions of the group.

    - **List groups**

      This permission allows user to list groups. This permission is necessary for use of other options from the **Groups** group.

    - **Create group**

      This permission allows user to create a new user group.

      If the user group is to be created in server web interface, the user needs to have the list groups permission (p. 126) to view a list of groups and to access this option.

    - **Delete group**

This permission allows user to delete a user group.

Only empty groups can be deleted. You need to have the list groups permission (p. 126) to view list of groups and to access this option.

- **Edit group**

  This permission allow user to edit user groups.

  This permission does not include **User assignment** and **Permission assignment**.

  If the user group is to be edited from server web interface, the user needs to have the list groups permission (p. 126).

- **Users assignment**

  This permission allows user to assign users to groups.

  The user needs Edit group permission (p. 127) to commit the changes in the assignment.

  If the assignment is to be edited in server web interface, the user needs to have the list groups permission (p. 126) to list the groups.

- **Permission assignment**

  This permission allows user to configure group **Permissions**.

  The user needs have the Edit group permission (p. 127) to commit the changes.

  If the permissions are to be edited in server web interface, the user needs to have the list groups permission (p. 126) to list the groups.

- **Secure parameters administration**

  - **Secure params**

    This permission allows user to change the value of a secure parameter.

    The user can use secure parameters in graphs even without this permission.

- **CloverETL/System info sections**

  This permission allows user to view **System Info** and **CloverETL Info** sections.

- **CloverETL Server properties**

  This permission allows user to view **Server Properties** tab and **Data Profiler properties** tab in **CloverETL Info** section.

  The user needs to have the CloverETL/System info sections permission (p. 127) to access **CloverETL Info** section.

- **Reload license**

  This permission allows user to reload and view the server license.

  The user needs to have the CloverETL/System info sections permission (p. 127) to access the **Configuration** section.

- **Upload license**

This permission allows user to update the server license.

The user needs to have the CloverETL/System info sections permission (p. 127) to access the **Configuration** section.

See CloverETL Server Activation (p. 45).

- **Server Configuration Management**

  This permission allows user to import and export the server configuration.

  See Chapter 17, Server Configuration Migration (p. 150).

  - **Export Server Configuration**

    This permission allows user to export the server configuration.

    See Server Configuration Export (p. 151).

  - **Import Server Configuration**

    This permission allows user to import the server configuration.

    See Server Configuration Import (p. 152).

- **Temp Space Management**

  This permission allows user to access **Temp Space Management** section.

  See Chapter 12, Temp Space Management (p. 101).

- **Server Setup**

  This permission allows user to access the server setup.

  See Chapter 7, Setup (p. 63).

- **Heap Memory Dump**

  This permission allows user to create a **Thread dump** and a **Heap Memory Dump**.

  See Chapter 18, Diagnostics (p. 156).

- **Groovy Code API**

  This permission allows user to run groovy scripts.

  See Groovy Code API (p. 241).

- **Open Profiler Reporting Console**

  This permission allows user to login to the **Profiler reporting console**.

  The permission is necessary to view the results of Clover Profiling Jobs in Designer.

  Even without this permission, a user can create and run .cpj jobs from Designer.

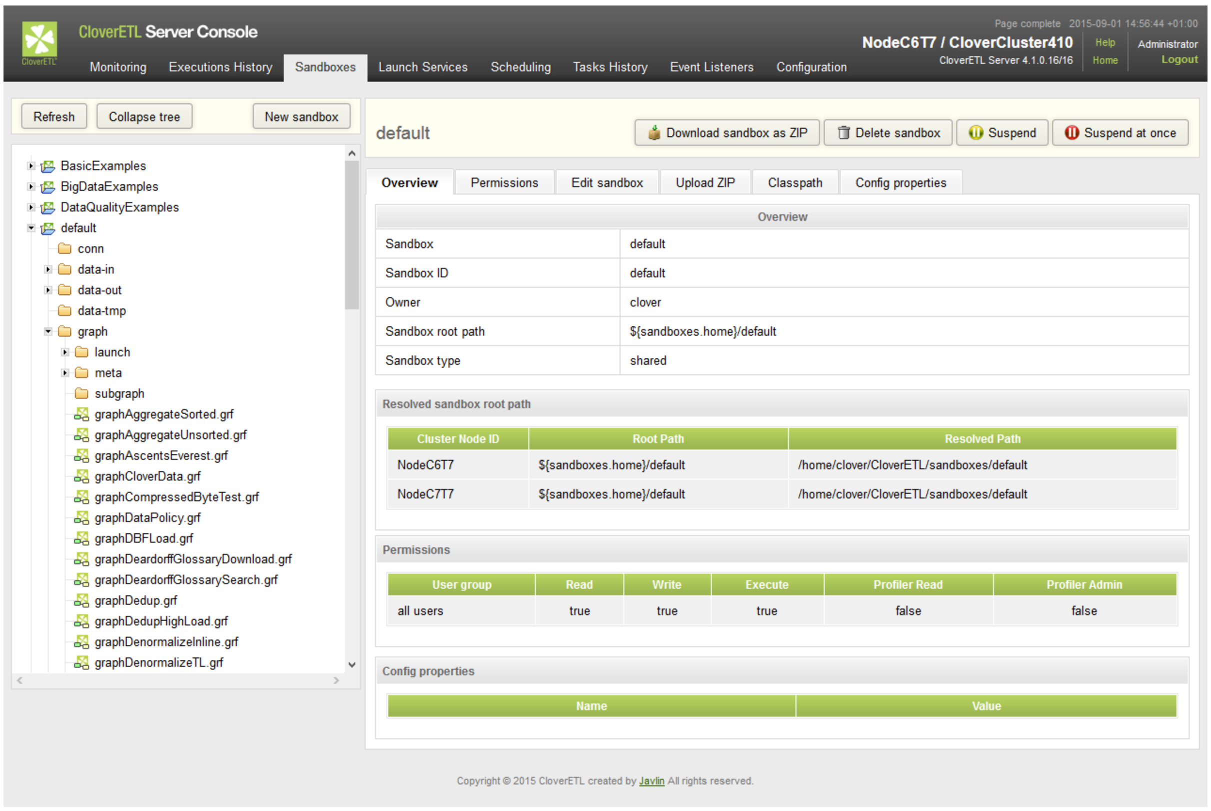
Chapter 15. Server Side Job Files - Sandboxes

A sandbox is a place 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 29, Clustering Features (p. 247)). 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.



- BasicExamples
- BigDataExamples
- DataQualityExamples
- default
  - conn
  - data-in
  - data-out
  - data-tmp
  - graph
    - launch
    - meta
    - subgraph
    - graphAggregateSorted.grf
    - graphAggregateUnsorted.grf
    - graphAscentsEverest.grf
    - graphCloverData.grf
    - graphCompressedByteTest.grf
    - graphDataPolicy.grf
    - graphDBFLoad.grf
    - graphDeardorffGlossaryDownload.grf
    - graphDeardorffGlossarySearch.grf
    - graphDedup.grf
    - graphDedupHighLoad.grf
    - graphDenormalizeInline.grf
    - graphDenormalizeTL.grf

Overview

Sandbox	default
Sandbox ID	default
Owner	clover
Sandbox root path	${sandboxes.home}/default
Sandbox type	shared

Resolved sandbox root path

Cluster Node ID	Root Path	Resolved Path
NodeC6T7	${sandboxes.home}/default	/home/clover/CloverETL/sandboxes/default
NodeC7T7	${sandboxes.home}/default	/home/clover/CloverETL/sandboxes/default

Permissions

User group	Read	Write	Execute	Profiler Read	Profiler Admin
all users	true	true	true	false	false

Config properties

Name	Value

Copyright © 2015 CloverETL created by Javlin All rights reserved.

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	A 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 ${sandboxes.home} placeholder, which may be configurable in the CloverETL Server configuration. So e.g. for the sandbox with ID "dataReports" the specified value of the "root path" would be "${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 dependent). Thus on the unix-like OS, the fully resolved sandbox root path may be: "/home/clover/CloverETL/sandboxes/dataReports". See Chapter 29, Clustering Features(p. 247) 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).

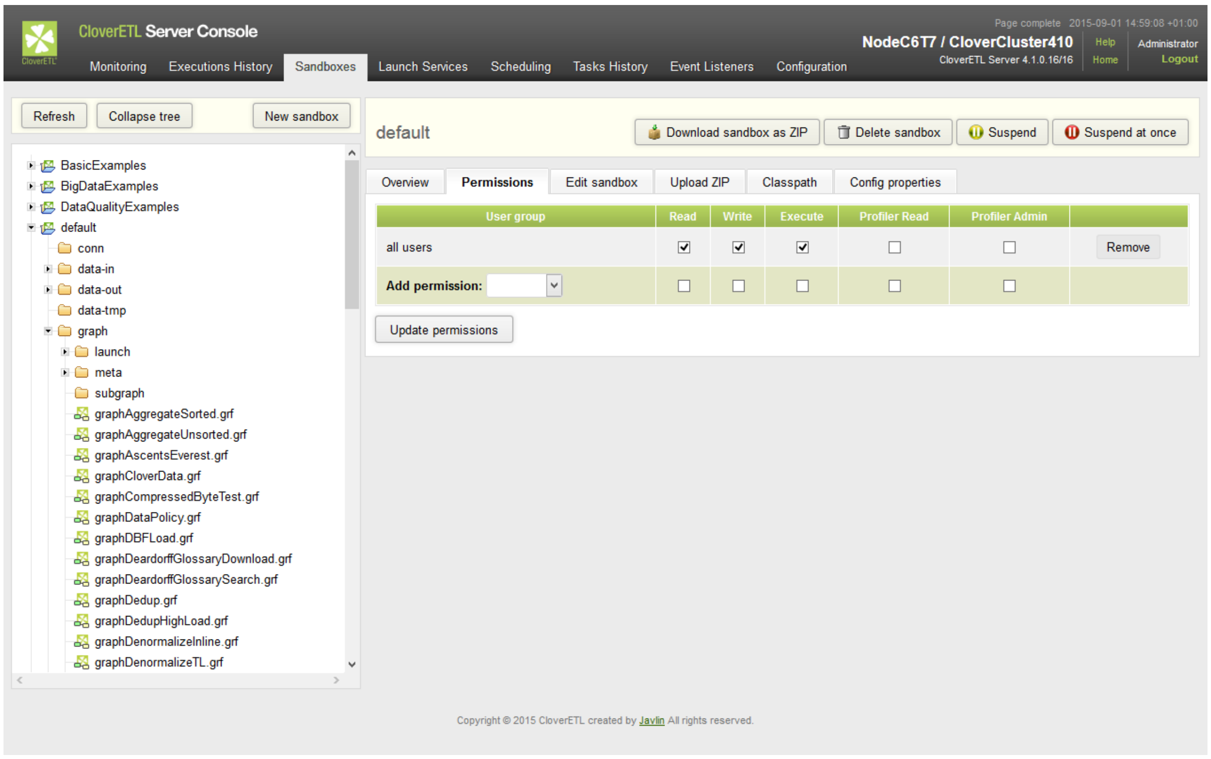
- sandbox:// URLs

  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. 252) 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.



User group	Read	Write	Execute	Profiler Read	Profiler Admin	
all users	☑	☑	☑	☐	☐	Remove
Add permission:	☐	☐	☐	☐	☐	

Figure 15.2. Sandbox Permissions in CloverETL Server Web GUI

Permissions to a 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:

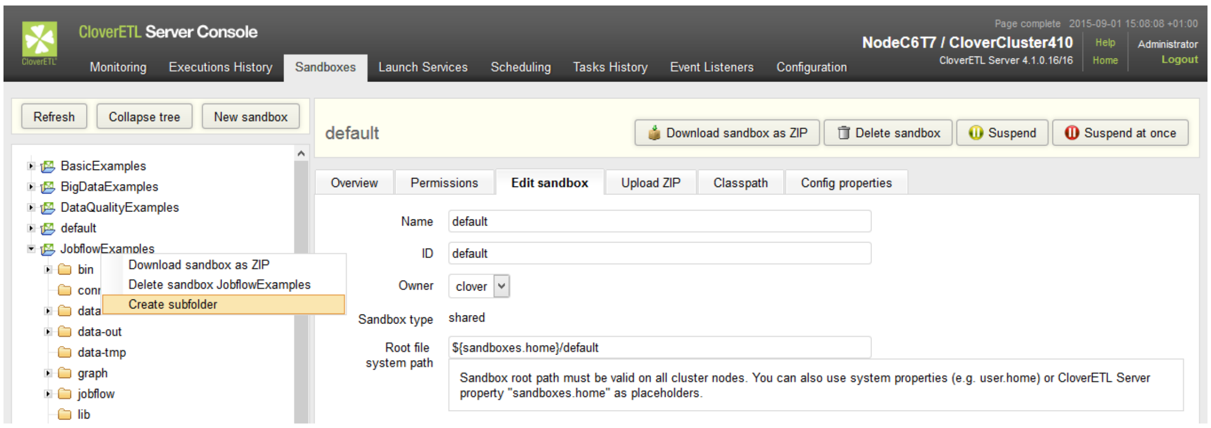
Table 15.2. Sandbox permissions

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 22, Scheduling (p. 183). 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. 110) 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".

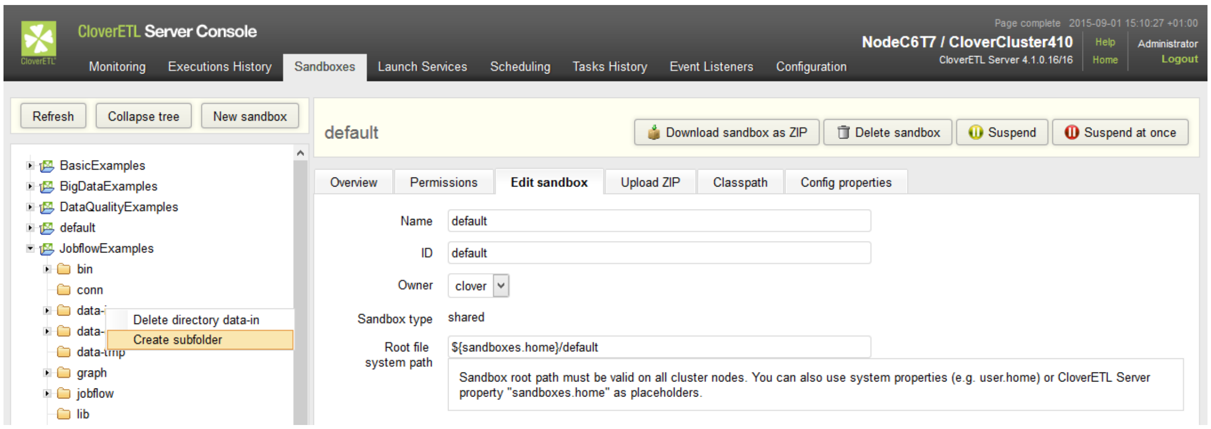


Name: default
ID: default
Owner: clover
Sandbox type: shared
Root file system path: ${sandboxes.home}/default

Sandbox root path must be valid on all cluster nodes. You can also use system properties (e.g. user.home) or CloverETL Server property "sandboxes.home" as placeholders.

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.



Name: default
ID: default
Owner: clover
Sandbox type: shared
Root file system path: ${sandboxes.home}/default

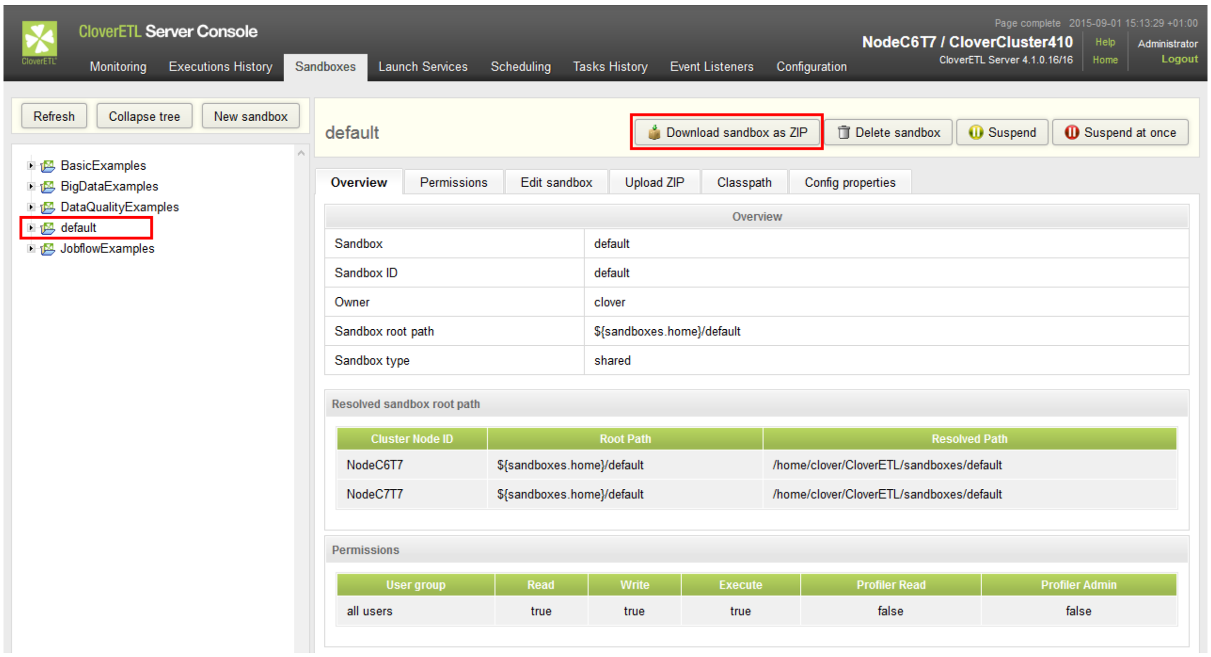
Sandbox root path must be valid on all cluster nodes. You can also use system properties (e.g. user.home) or CloverETL Server property "sandboxes.home" as placeholders.

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

Download sandbox as ZIP

Select a 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.



default

Overview

Sandbox	default
Sandbox ID	default
Owner	clover
Sandbox root path	${sandboxes.home}/default
Sandbox type	shared

Resolved sandbox root path

Cluster Node ID	Root Path	Resolved Path
NodeC6T7	${sandboxes.home}/default	/home/clover/CloverETL/sandboxes/default
NodeC7T7	${sandboxes.home}/default	/home/clover/CloverETL/sandboxes/default

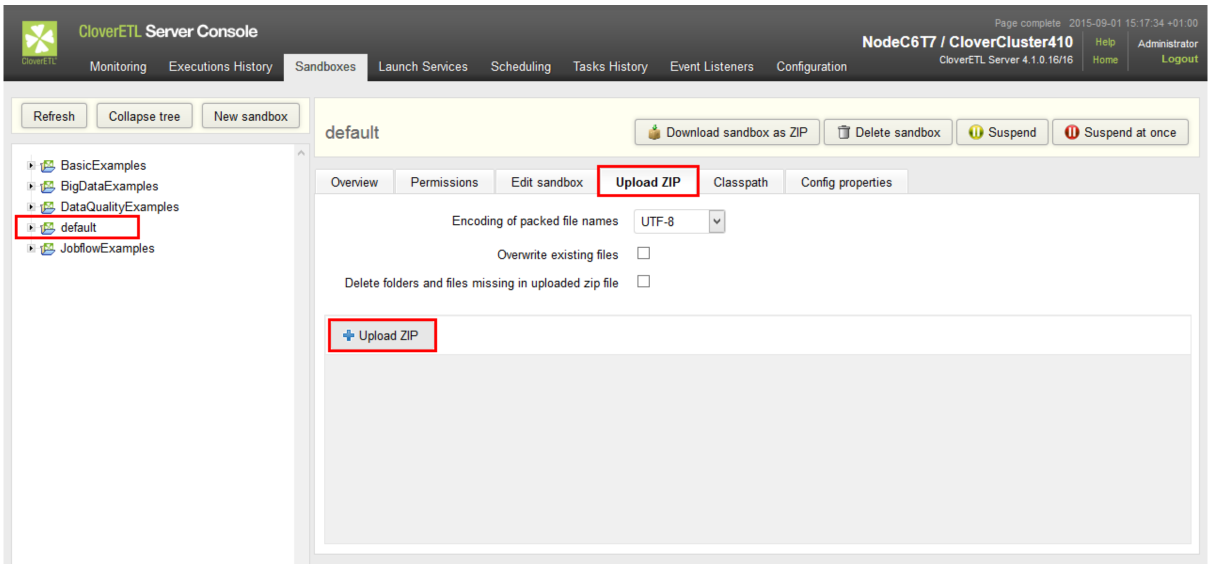
Permissions

User group	Read	Write	Execute	Profiler Read	Profiler Admin
all users	true	true	true	false	false

Figure 15.5. Web GUI - download sandbox as ZIP

Upload ZIP to sandbox

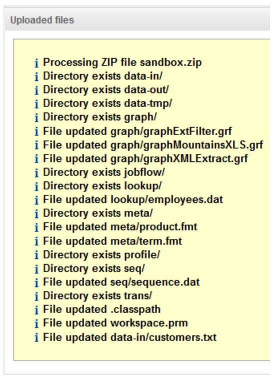
Select a sandbox in left panel. You must have write permission to the selected sandbox. Then select tab "Upload ZIP" in the right panel. Upload of a ZIP is parametrized by couple of switches, which are described below. Open a common file chooser dialog by button "+ Upload ZIP". When you choose a ZIP 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.



default

Encoding of packed file names	UTF-8
Overwrite existing files	☐
Delete folders and files missing in uploaded zip file	☐

Figure 15.6. Web GUI - upload ZIP to sandbox


Uploaded files

- Processing ZIP file sandbox.zip
- Directory exists data-in/
- Directory exists data-out/
- Directory exists data-tmp/
- Directory exists graph/
- File updated graph/graphExtFilter.grf
- File updated graph/graphMountainsXLS.grf
- File updated graph/graphXMLExtract.grf
- Directory exists jobflow/
- Directory exists lookup/
- File updated lookup/employees.dat
- Directory exists meta/
- File updated meta/product.fmt
- File updated meta/term.fmt
- Directory exists profile/
- Directory exists seq/
- File updated seq/sequence.dat
- Directory exists trans/
- File updated .classpath
- File updated workspace.prm
- File updated data-in/customers.txt

Figure 15.7. Web GUI - upload ZIP results

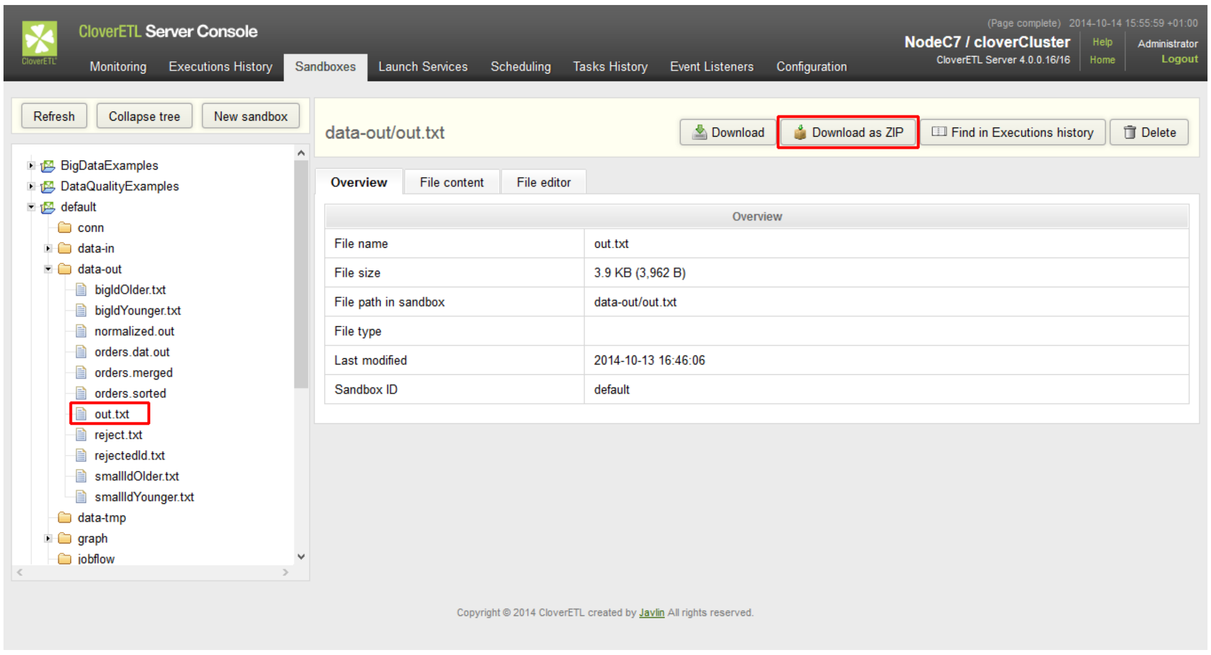
Table 15.3. ZIP upload parameters

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 overwritten by a 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 a file in the 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.



Overview

File name	out.txt
File size	3.9 KB (3,962 B)
File path in sandbox	data-out/out.txt
File type	
Last modified	2014-10-13 16:46:06
Sandbox ID	default

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 nor 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. 60) 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}`.

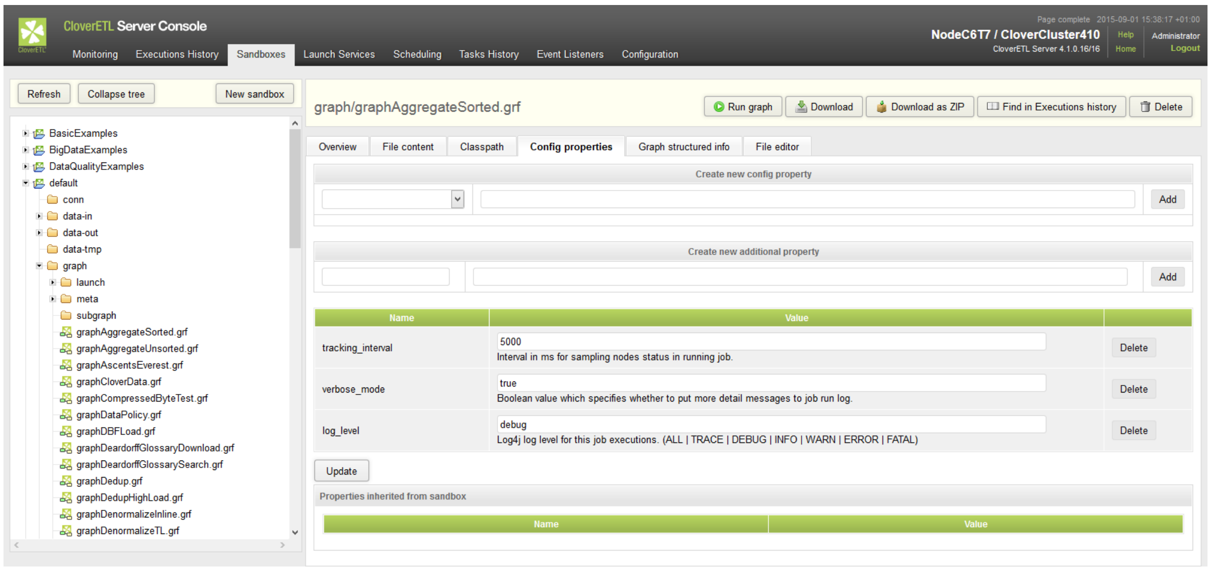
In path definition, you can use system properties - e.g. `${java.io.tmpdir}` - and some of server config properties: `${sandboxes.home}`, `${sandboxes.home.partitioned}` and `${sandboxes.home.local}`.

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. In cluster environment, the limit is per node.
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 | 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. 98) for details about log4j configuration.
max_graph_instance_age	0	A time interval in ms which specifies how long may a transformation instance last in server's cache. 0 means that the transformation is initialized and released for each execution. The transformation cannot be stored in the pool and reused in some cases (a 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

Property name	Default value	Description
		"DEFAULT_PATH_SEPARATOR_REGEX". Directory path must always end with a 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". The directory path must always end with a 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 a 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 a dictionary. Another Reformat component get 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

Property name	Default value	Description
		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 the 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.



Name	Value	
tracking_interval	5000 Interval in ms for sampling nodes status in running job.	Delete
verbose_mode	true Boolean value which specifies whether to put more detail messages to job run log.	Delete
log_level	debug Log4j log level for this job executions. (ALL | TRACE | DEBUG | INFO | WARN | ERROR | FATAL)	Delete

Name	Value

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. Unfortunately, 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. 60) 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 environment. In case of standalone server, the server-view is the same as node detail in cluster environment.

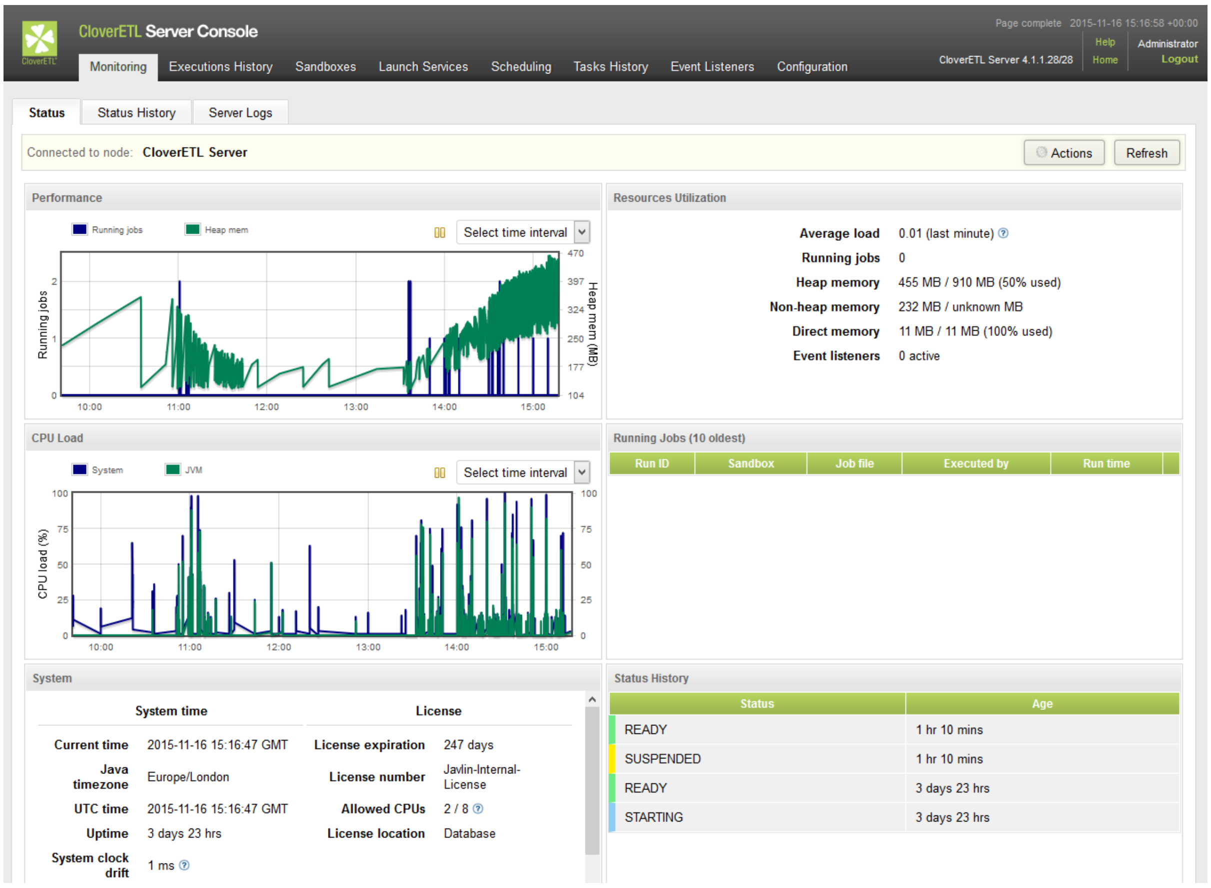
The section is refreshed each 15 seconds so the displayed data is up-to-date. Page can be also anytime refreshed manually by the "Refresh" button

Standalone Server Detail

Standalone server detail view displays info collected from the standalone server. The info is grouped in several panels. The following ones are displayed by default.

- Performance
- Resource utilization
- 10 longest-running jobs
- System
- Status history

You can display the hidden actions with **Actions** button: choose **Actions** →**Show details**.



Average load	0.01 (last minute)
Running jobs	0
Heap memory	455 MB / 910 MB (50% used)
Non-heap memory	232 MB / unknown MB
Direct memory	11 MB / 11 MB (100% used)
Event listeners	0 active

Run ID	Sandbox	Job file	Executed by	Run time

System time		License	
Current time	2015-11-16 15:16:47 GMT	License expiration	247 days
Java timezone	Europe/London	License number	Javlin-Internal-License
UTC time	2015-11-16 15:16:47 GMT	Allowed CPUs	2 / 8
Uptime	3 days 23 hrs	License location	Database
System clock drift	1 ms		

Status	Age
READY	1 hr 10 mins
SUSPENDED	1 hr 10 mins
READY	3 days 23 hrs
STARTING	3 days 23 hrs

Figure 16.1. Standalone server detail

Performance

The Performance panel contains a chart with two basic performance statistics: a number of running jobs and an amount of used heap memory. The graph displays values gathered within a specific interval. The interval can be set up with the combo box above the graph or it can be configured by "cluster.node.sendinfo.history.interval" config property. By default, the values are gathered within a couple of last minutes.

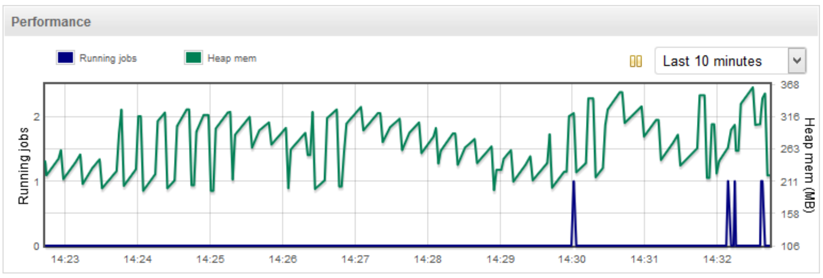
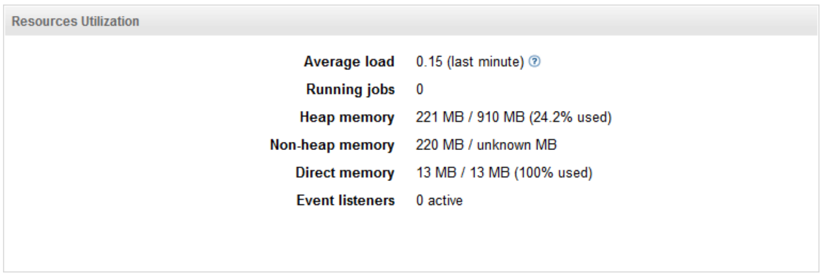


Figure 16.2. Performance

Resource Utilization

Resource utilization shows stats: average load, number of runnig jobs, heap memory, non-heap memory, and event listeners.



Resources Utilization

Average load	0.15 (last minute)
Running jobs	0
Heap memory	221 MB / 910 MB (24.2% used)
Non-heap memory	220 MB / unknown MB
Direct memory	13 MB / 13 MB (100% used)
Event listeners	0 active

Figure 16.3. Resource Utilization

CPU Load

The CPU Load panel displays a chart with info about total CPU load and CPU load caused by JVM.

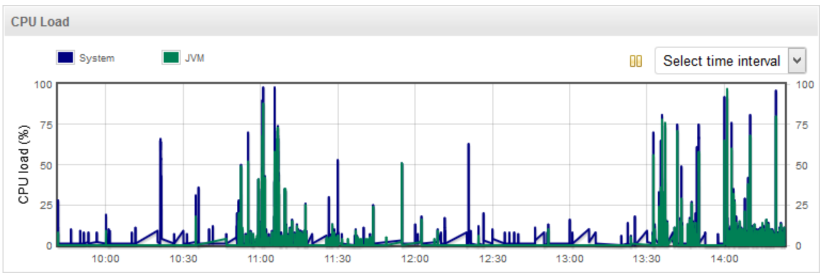
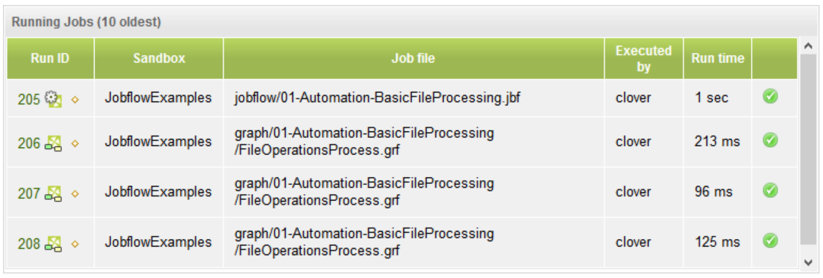


Figure 16.4. CPU Load

Running Jobs

Running jobs panel lists currently running jobs, 10 oldest runs are displayed.



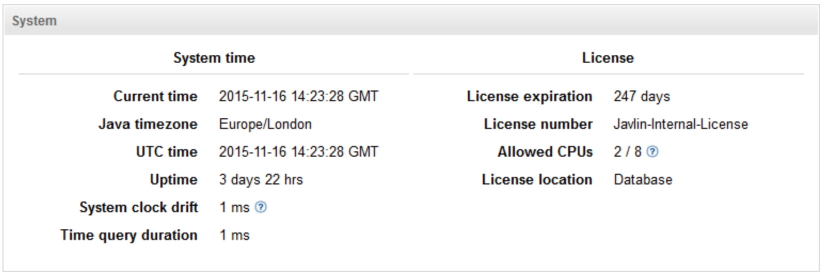
Running Jobs (10 oldest)

Run ID	Sandbox	Job file	Executed by	Run time	
205	JobflowExamples	jobflow/01-Automation-BasicFileProcessing.jbf	clover	1 sec	✓
206	JobflowExamples	graph/01-Automation-BasicFileProcessing /FileOperationsProcess.grf	clover	213 ms	✓
207	JobflowExamples	graph/01-Automation-BasicFileProcessing /FileOperationsProcess.grf	clover	96 ms	✓
208	JobflowExamples	graph/01-Automation-BasicFileProcessing /FileOperationsProcess.grf	clover	125 ms	✓

Figure 16.5. Running jobs

System

System panel contains info about operating system and license.



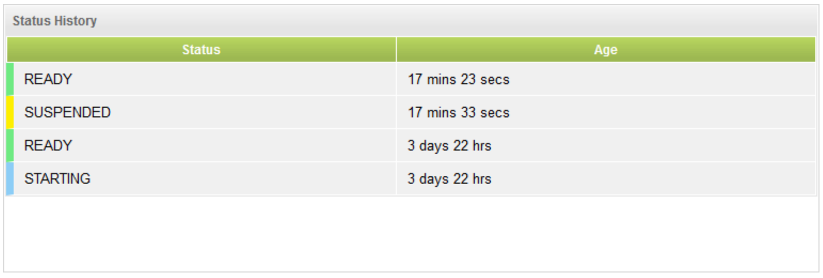
System

System time		License	
Current time	2015-11-16 14:23:28 GMT	License expiration	247 days
Java timezone	Europe/London	License number	Javlin-Internal-License
UTC time	2015-11-16 14:23:28 GMT	Allowed CPUs	2 / 8
Uptime	3 days 22 hrs	License location	Database
System clock drift	1 ms		
Time query duration	1 ms		

Figure 16.6. System

Status History

Status history panel displays node statuses history since restart of the server.



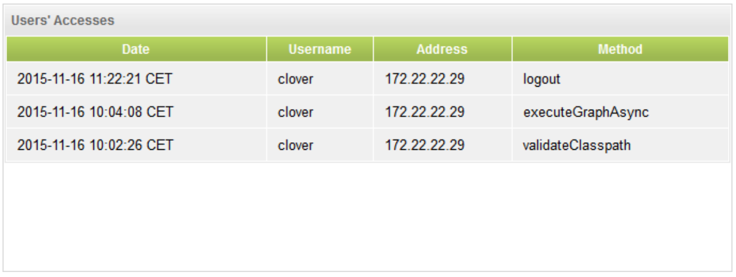
Status History

Status	Age
READY	17 mins 23 secs
SUSPENDED	17 mins 33 secs
READY	3 days 22 hrs
STARTING	3 days 22 hrs

Figure 16.7. Status History

User's Access

User's Access panel lists info about activities on files performed by users. The list displays a timestamp of an event, a username, and a name of the method.



Users' Accesses

Date	Username	Address	Method
2015-11-16 11:22:21 CET	clover	172.22.22.29	logout
2015-11-16 10:04:08 CET	clover	172.22.22.29	executeGraphAsync
2015-11-16 10:02:26 CET	clover	172.22.22.29	validateClasspath

Figure 16.8. Users' Accesses panel

Classloader cache

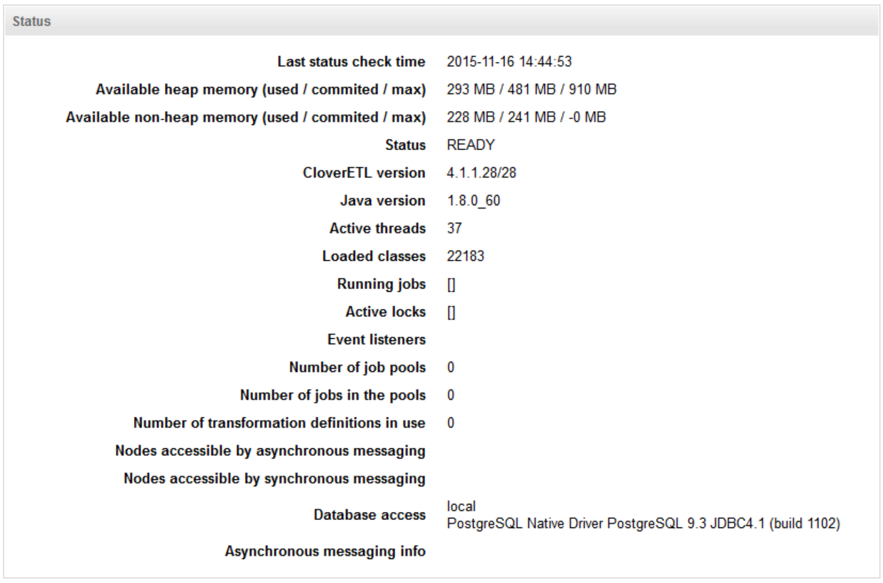
Classloader cache lists all currently cached classloaders. The classloader cache may be empty as classloader caching is disabled by default.

Classloader cache (empty)

Figure 16.9. Classloader cache

Status

Status panel displays current node status since last server restart. It displays current server status (ready, stopped, ...), exact Java version, exact CloverETL Server version, way of access to database, URLs for synchronous and asynchronous messaging, available heap and non-heap memory, etc.



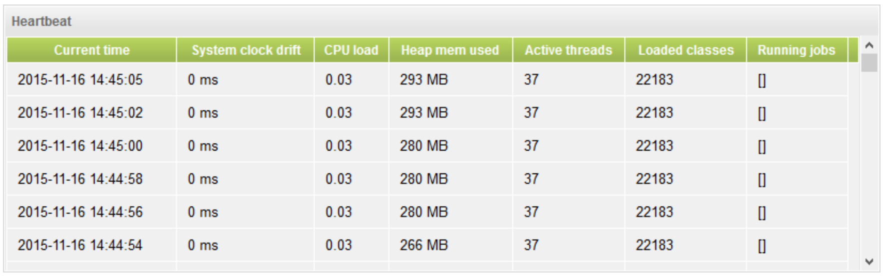
Status

Last status check time	2015-11-16 14:44:53
Available heap memory (used / commited / max)	293 MB / 481 MB / 910 MB
Available non-heap memory (used / commited / max)	228 MB / 241 MB / -0 MB
Status	READY
CloverETL version	4.1.1.28/28
Java version	1.8.0_60
Active threads	37
Loaded classes	22183
Running jobs	[]
Active locks	[]
Event listeners	
Number of job pools	0
Number of jobs in the pools	0
Number of transformation definitions in use	0
Nodes accessible by asynchronous messaging	
Nodes accessible by synchronous messaging	
Database access	local PostgreSQL Native Driver PostgreSQL 9.3 JDBC4.1 (build 1102)
Asynchronous messaging info	

Figure 16.10. Status

Heartbeat

Heartbeat panel displays a list of heartbeat events and their results.



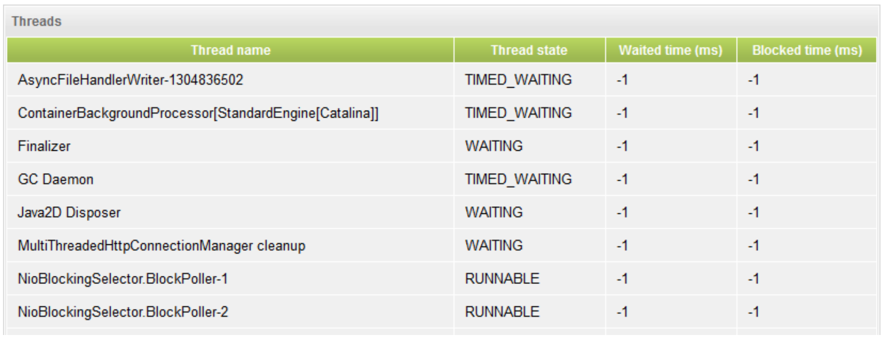
Heartbeat

Current time	System clock drift	CPU load	Heap mem used	Active threads	Loaded classes	Running jobs
2015-11-16 14:45:05	0 ms	0.03	293 MB	37	22183	[]
2015-11-16 14:45:02	0 ms	0.03	293 MB	37	22183	[]
2015-11-16 14:45:00	0 ms	0.03	280 MB	37	22183	[]
2015-11-16 14:44:58	0 ms	0.03	280 MB	37	22183	[]
2015-11-16 14:44:56	0 ms	0.03	280 MB	37	22183	[]
2015-11-16 14:44:54	0 ms	0.03	266 MB	37	22183	[]

Figure 16.11. Heartbeat

Threads

Threads panel lists java threads and their states.



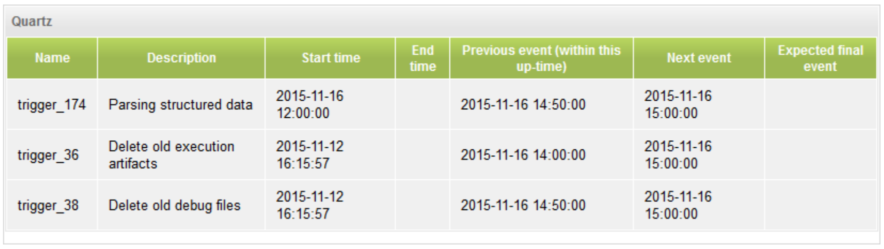
Threads

Thread name	Thread state	Waited time (ms)	Blocked time (ms)
AsyncFileHandlerWriter-1304836502	TIMED_WAITING	-1	-1
ContainerBackgroundProcessor[StandardEngine[Catalina]]	TIMED_WAITING	-1	-1
Finalizer	WAITING	-1	-1
GC Daemon	TIMED_WAITING	-1	-1
Java2D Disposer	WAITING	-1	-1
MultiThreadedHttpConnectionManager cleanup	WAITING	-1	-1
NioBlockingSelector.BlockPoller-1	RUNNABLE	-1	-1
NioBlockingSelector.BlockPoller-2	RUNNABLE	-1	-1

Figure 16.12. Threads

Quartz

Quartz panel lists scheduled actions: their name, description, start time, end time, time of previous event, time of next event and expected final event.



Quartz

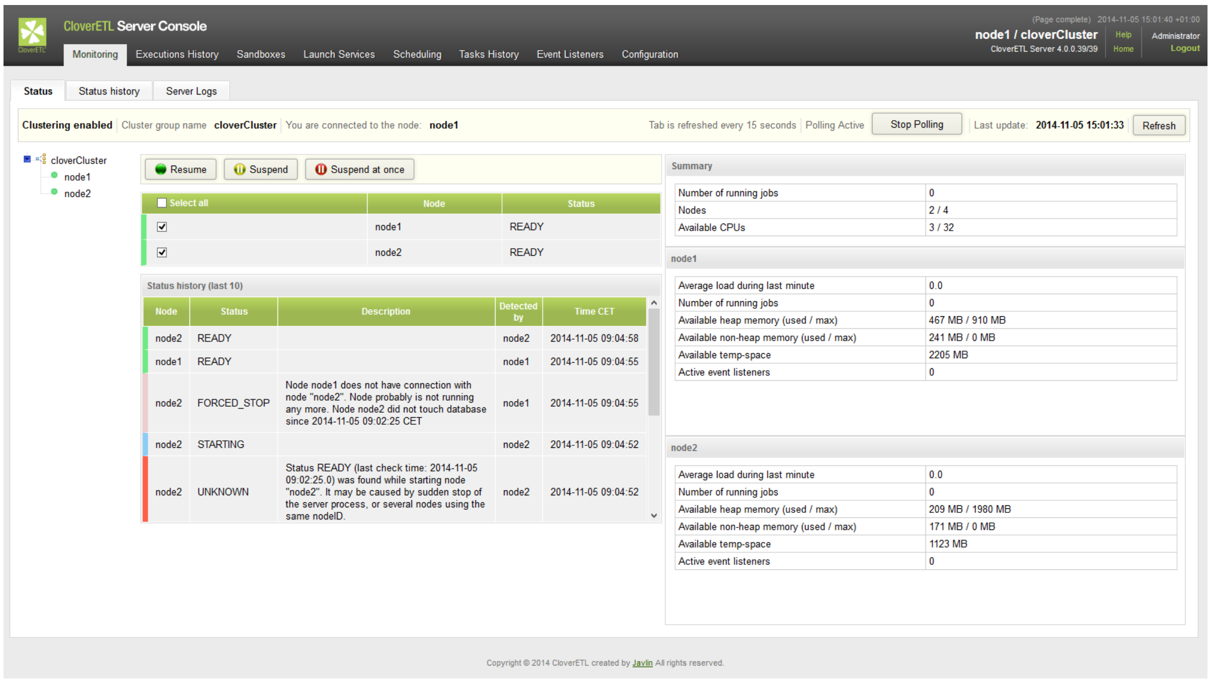
Name	Description	Start time	End time	Previous event (within this up-time)	Next event	Expected final event
trigger_174	Parsing structured data	2015-11-16 12:00:00		2015-11-16 14:50:00	2015-11-16 15:00:00	
trigger_36	Delete old execution artifacts	2015-11-12 16:15:57		2015-11-16 14:00:00	2015-11-16 15:00:00	
trigger_38	Delete old debug files	2015-11-12 16:15:57		2015-11-16 14:50:00	2015-11-16 15:00:00	

Figure 16.13. Quartz

Cluster Overview

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

- List of nodes with a 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.



Clustering enabled | Cluster group name cloverCluster | You are connected to the node: node1

Tab is refreshed every 15 seconds | Polling Active | Last update: 2014-11-05 15:01:33

Select all	Node	Status
✓	node1	READY
✓	node2	READY

Status history (last 10)

Node	Status	Description	Detected by	Time CET
node2	READY		node2	2014-11-05 09:04:58
node1	READY		node1	2014-11-05 09:04:55
node2	FORCED_STOP	Node node1 does not have connection with node "node2". Node probably is not running any more. Node node2 did not touch database since 2014-11-05 09:02:25 CET	node1	2014-11-05 09:04:55
node2	STARTING		node2	2014-11-05 09:04:52
node2	UNKNOWN	Status READY (last check time: 2014-11-05 09:02:25.0) was found while starting node "node2". It may be caused by sudden stop of the server process, or several nodes using the same nodeID.	node2	2014-11-05 09:04:52

Summary

Number of running jobs	0
Nodes	2 / 4
Available CPUs	3 / 32

node1

Average load during last minute	0.0
Number of running jobs	0
Available heap memory (used / max)	467 MB / 910 MB
Available non-heap memory (used / max)	241 MB / 0 MB
Available temp-space	2205 MB
Active event listeners	0

node2

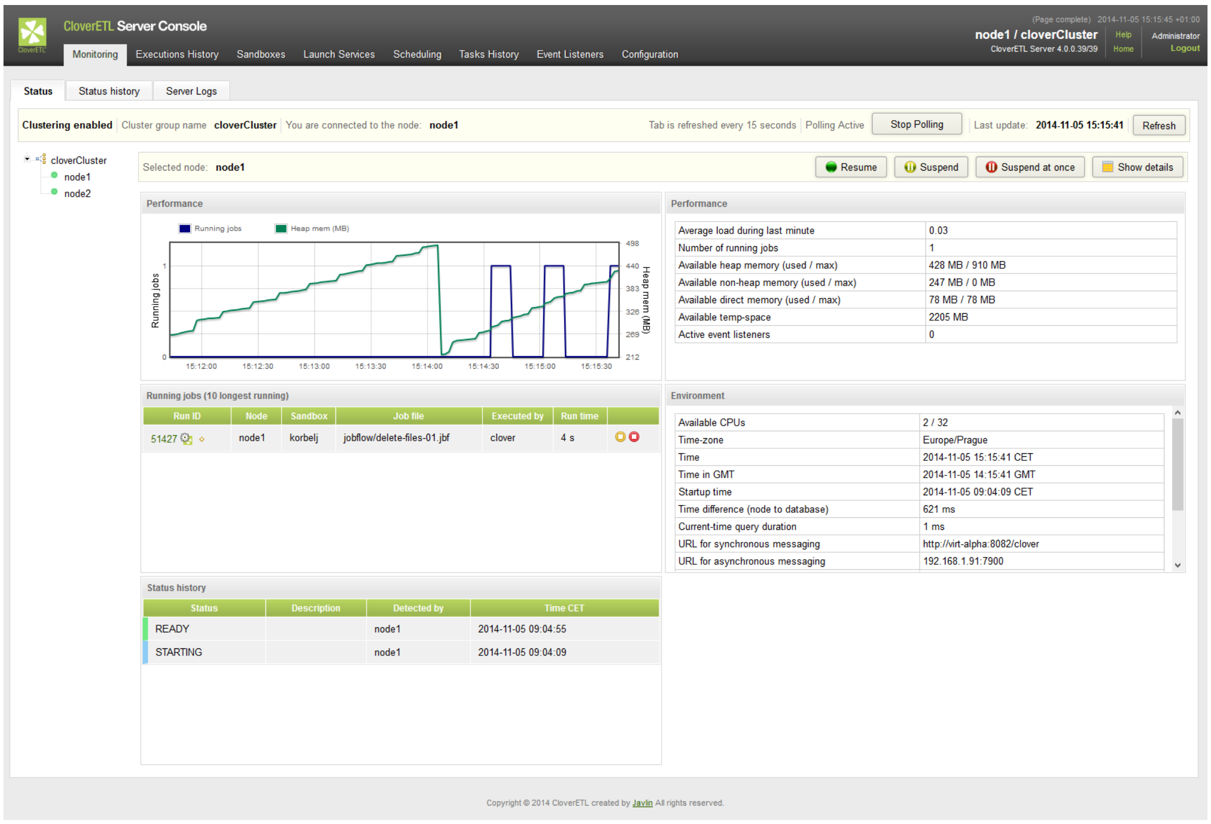
Average load during last minute	0.0
Number of running jobs	0
Available heap memory (used / max)	209 MB / 1980 MB
Available non-heap memory (used / max)	171 MB / 0 MB
Available temp-space	1123 MB
Active event listeners	0

Copyright © 2014 CloverETL created by Javlin All rights reserved.

Figure 16.14. Cluster overview

Node Detail

Node Detail is similar to the "Standalone server detail" mentioned above, however it displays detail info about node selected in the tree on the left.



Average load during last minute	0.03
Number of running jobs	1
Available heap memory (used / max)	428 MB / 910 MB
Available non-heap memory (used / max)	247 MB / 0 MB
Available direct memory (used / max)	78 MB / 78 MB
Available temp-space	2205 MB
Active event listeners	0

Run ID	Node	Sandbox	Job file	Executed by	Run time	
51427	node1	korbelj	jobflow/delete-files-01.jbf	clover	4 s	

Available CPUs	2 / 32
Time-zone	Europe/Prague
Time	2014-11-05 15:15:41 CET
Time in GMT	2014-11-05 14:15:41 GMT
Startup time	2014-11-05 09:04:09 CET
Time difference (node to database)	621 ms
Current-time query duration	1 ms
URL for synchronous messaging	http://virt-alpha:8082/clover
URL for asynchronous messaging	192.168.1.91:7900

Status	Description	Detected by	Time CET
READY		node1	2014-11-05 09:04:55
STARTING		node1	2014-11-05 09:04:09

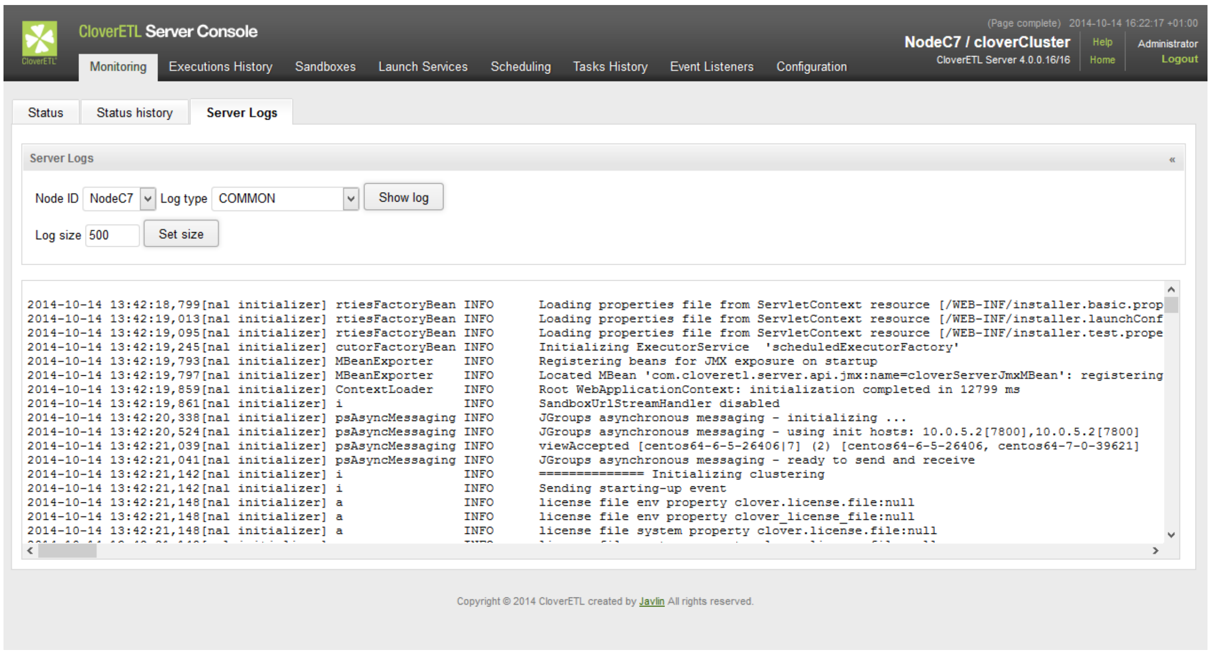
Figure 16.15. Node detail

Server Logs

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

There are 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. 99) for details
- USER_ACTION - Contains some of user operations, e.g. login, logout, job execution



Node ID NodeC7 Log type COMMON

Log size 500

```
2014-10-14 13:42:18,799[nal initializer] rtiesFactoryBean INFO      Loading properties file from ServletContext resource [/WEB-INF/installer.basic.prop
2014-10-14 13:42:19,013[nal initializer] rtiesFactoryBean INFO      Loading properties file from ServletContext resource [/WEB-INF/installer.launchConf
2014-10-14 13:42:19,095[nal initializer] rtiesFactoryBean INFO      Loading properties file from ServletContext resource [/WEB-INF/installer.test.prope
2014-10-14 13:42:19,245[nal initializer] cutorFactoryBean INFO      Initializing ExecutorService  'scheduledExecutorFactory'
2014-10-14 13:42:19,793[nal initializer] MBeanExporter    INFO      Registering beans for JMX exposure on startup
2014-10-14 13:42:19,797[nal initializer] MBeanExporter    INFO      Located MBean 'com.cloveretl.server.api.jmx:name=cloverServerJmxMBean': registering
2014-10-14 13:42:19,859[nal initializer] ContextLoader    INFO      Root WebApplicationContext: initialization completed in 12799 ms
2014-10-14 13:42:19,861[nal initializer] i                INFO      SandboxUrlStreamHandler disabled
2014-10-14 13:42:20,338[nal initializer] psAsyncMessaging INFO      JGroups asynchronous messaging - initializing ...
2014-10-14 13:42:20,524[nal initializer] psAsyncMessaging INFO      JGroups asynchronous messaging - using init hosts: 10.0.5.2[7800],10.0.5.2[7800]
2014-10-14 13:42:21,039[nal initializer] psAsyncMessaging INFO      viewAccepted [centos64-6-5-26406|7] (2) [centos64-6-5-26406, centos64-7-0-39621]
2014-10-14 13:42:21,041[nal initializer] psAsyncMessaging INFO      JGroups asynchronous messaging - ready to send and receive
2014-10-14 13:42:21,142[nal initializer] i                INFO      ============== Initializing clustering
2014-10-14 13:42:21,142[nal initializer] i                INFO      Sending starting-up event
2014-10-14 13:42:21,148[nal initializer] a                INFO      license file env property clover.license.file:null
2014-10-14 13:42:21,148[nal initializer] a                INFO      license file env property clover_license_file:null
2014-10-14 13:42:21,148[nal initializer] a                INFO      license file system property clover.license.file:null
```

Copyright © 2014 CloverETL created by Javlin All rights reserved.

Figure 16.16. 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 manually 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 the 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. 110))
- Sandboxes (see Chapter 15, Server Side Job Files - Sandboxes (p. 129))
- Job Parameters (see Chapter 19, Graph/Jobflow Parameters (p. 158))
- Schedules (see Chapter 22, Scheduling (p. 183))
- Event Listeners (see Graph Event Listeners (p. 195), Jobflow Event Listeners (p. 202), JMS Messages Listeners (p. 205), File Event Listeners (remote and local) (p. 212))
- Launch Services (see Launch Services (p. 234))
- Temp Spaces (see Chapter 12, Temp Space Management (p. 101))

Permissions for Configuration Migration

Whether a user is entitled to perform configuration migration is determined by having *Server Configuration Management* permission; this permission has two sub-permissions: *Export Server Configuration* and *Import Server Configuration* (see Groups permissions (p. 118) for further information on permissions). These permissions are of higher priority than permissions related to a 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.

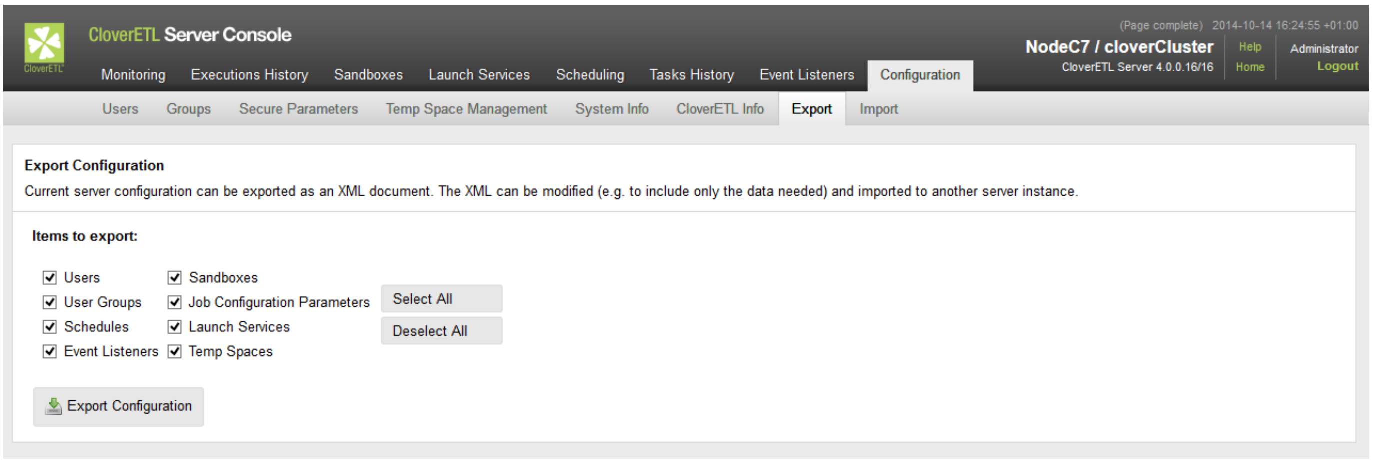
See Server Configuration permission (p. 128).

Server Configuration Export

Export of a 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. 151)). After clicking on the **Export Configuration** an XML file will be offered for download. The name of the XML file reflects time when the configuration was exported.

In case user manually edits the exported XML file, you shell ensure that the file has a valid content. This can be done by validation against XSD schema. The schema for a 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 the import to another server instance - for example to import schedules only.



Export Configuration

Current server configuration can be exported as an XML document. The XML can be modified (e.g. to include only the data needed) and imported to another server instance.

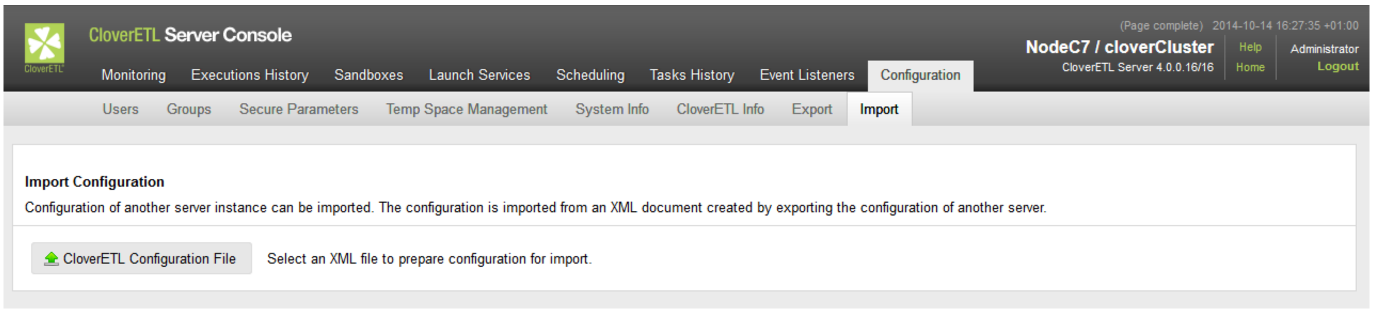
Items to export:

☑ Users	☑ Sandboxes
☑ User Groups	☑ Job Configuration Parameters
☑ Schedules	☑ Launch Services
☑ Event Listeners	☑ Temp Spaces

Figure 17.1. Server Configuration Export screen

Server Configuration Import

Import of CloverETL Server configuration merges the 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. 151). Import of server configuration is performed from the Server Console - the screen for import can be found in **Configuration** > **Import** section.



Import Configuration

Configuration of another server instance can be imported. The configuration is imported from an XML document created by exporting the configuration of another server.

Select an XML file to prepare configuration for 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:

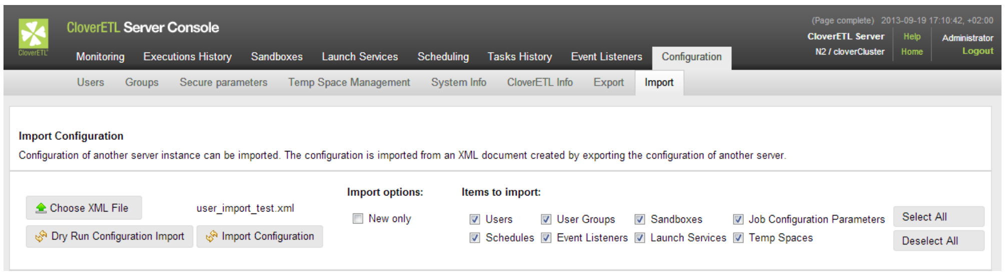
- users - user code
- user groups - group code
- sandboxes - sandbox code
- job parameters - triplet (job parameter name, sandbox code, job file)
- event listeners - event listener name
- schedule - schedule description
- launch service - triplet (service name, server user, service user group)
- temp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



(Page complete) 2013-09-19 17:10:42, +02:00
CloverETL Server
N2 / cloverCluster

Import Configuration

Configuration of another server instance can be imported. The configuration is imported from an XML document created by exporting the configuration of another server.

user_import_test.xml

Import options: ☐ New only

Items to import: ☑ Users ☑ User Groups ☑ Sandboxes ☑ Job Configuration Parameters ☑ Schedules ☑ Event Listeners ☑ Launch Services ☑ Temp Spaces

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 is 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. 154)). 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

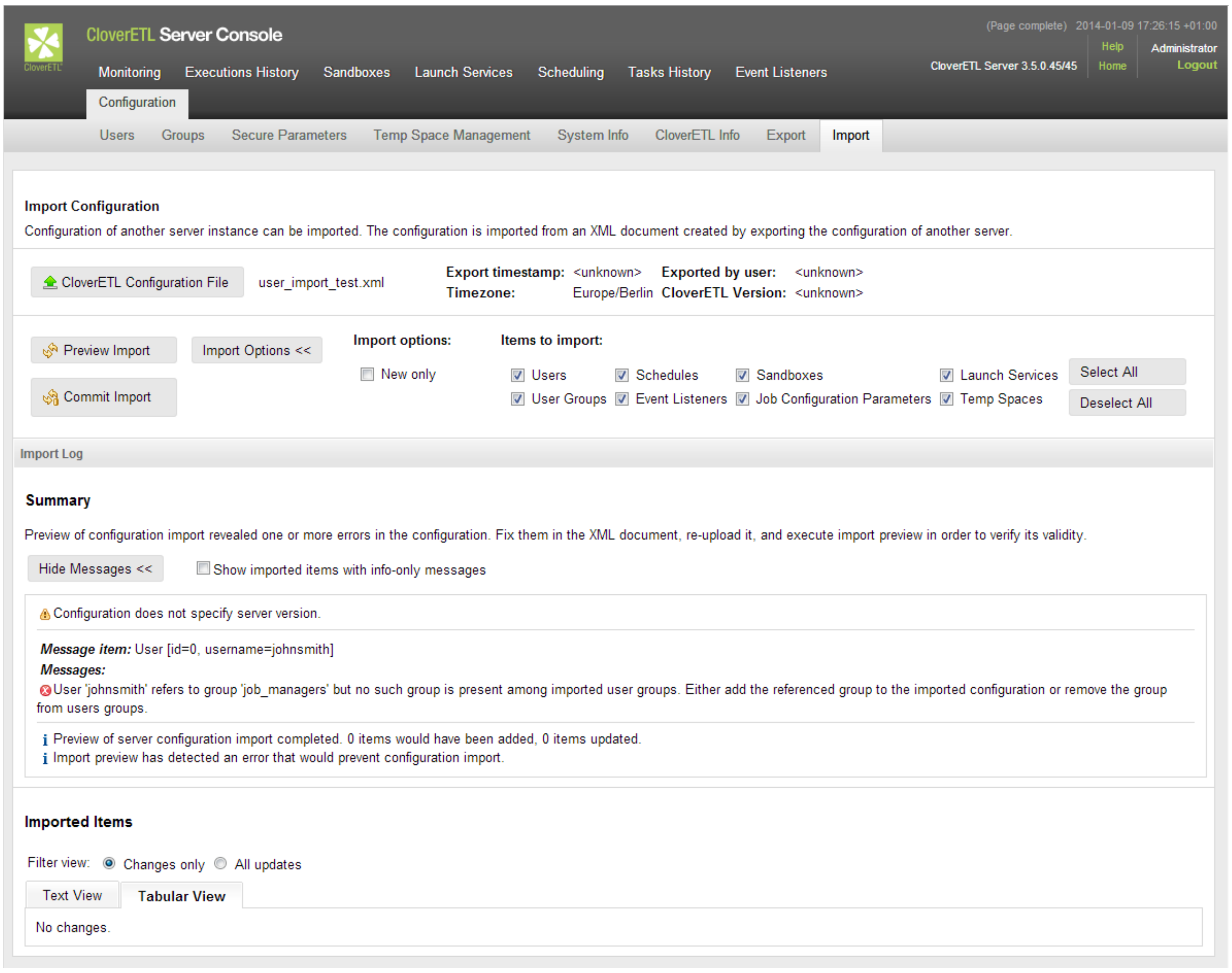
For the updated items, the state of the item before update is shown in the lower of the rows with less contrast text color, the new values of item's properties are shown in upper of the rows with regular text color. Actual changes are highlighted by background color change on respective property and 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 identical 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>
```



Import Configuration

Configuration of another server instance can be imported. The configuration is imported from an XML document created by exporting the configuration of another server.

CloverETL Configuration File   user_import_test.xml

Export timestamp: <unknown>   Exported by user: <unknown>
Timezone: Europe/Berlin   CloverETL Version: <unknown>

Import options: ☐ New only

Items to import: ☑ Users ☑ Schedules ☑ Sandboxes ☑ Launch Services ☑ User Groups ☑ Event Listeners ☑ Job Configuration Parameters ☑ Temp Spaces

Summary

Preview of configuration import revealed one or more errors in the configuration. Fix them in the XML document, re-upload it, and execute import preview in order to verify its validity.

☐ Show imported items with info-only messages

Configuration does not specify server version.

Message item: User [id=0, username=johnsmith]
Messages:
User 'johnsmith' refers to group 'job_managers' but no such group is present among imported user groups. Either add the referenced group to the imported configuration or remove the group from users groups.

Preview of server configuration import completed. 0 items would have been added, 0 items updated.
Import preview has detected an error that would prevent configuration import.

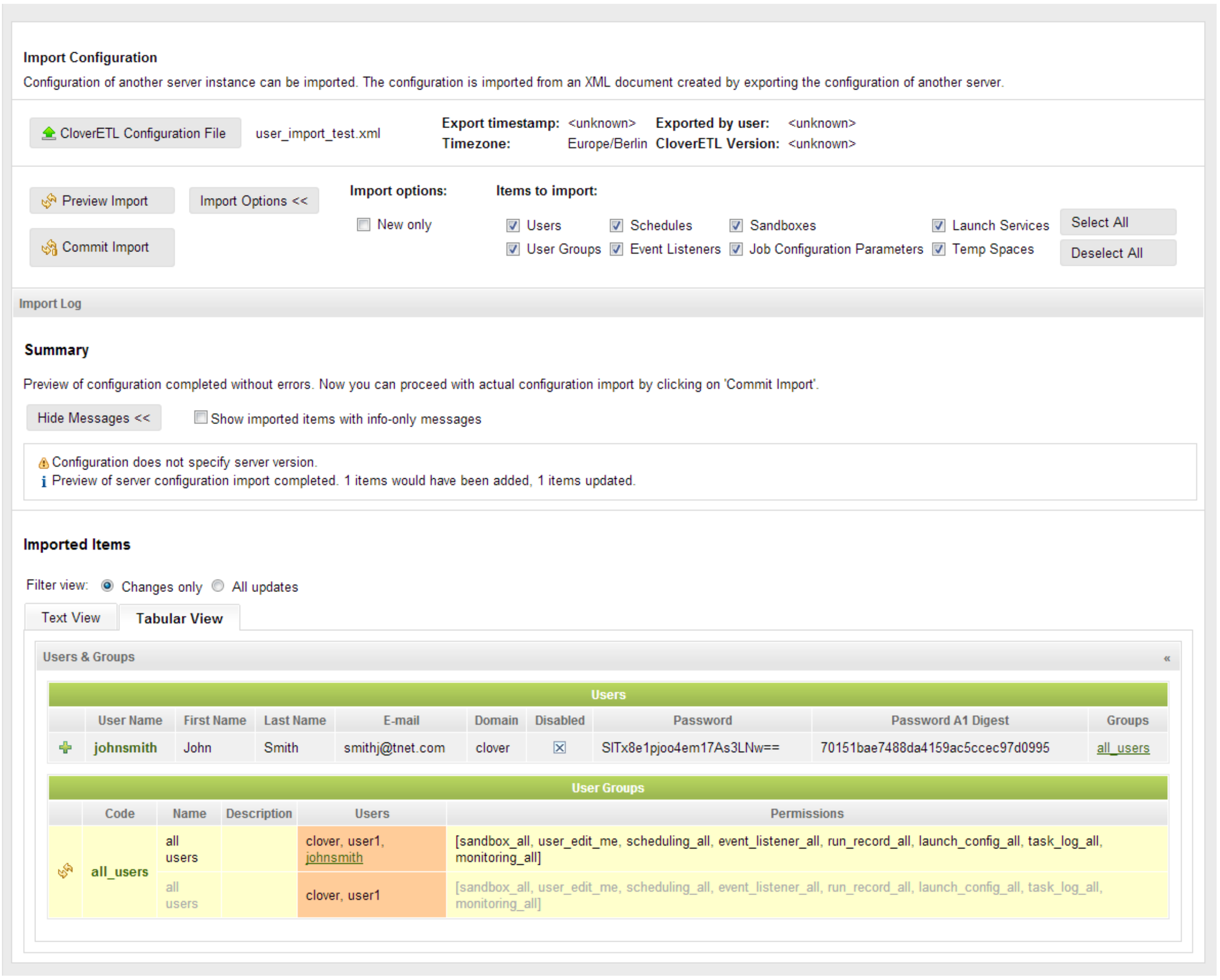
Imported Items

Filter view: [✓ Changes only] All updates

No changes.

Figure 17.4. Outcome of the import preview for configuration from Example 17.1 (p. 153)

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. 154)) .



Import Configuration

Configuration of another server instance can be imported. The configuration is imported from an XML document created by exporting the configuration of another server.

CloverETL Configuration File user_import_test.xml

Export timestamp: <unknown> **Exported by user:** <unknown>
Timezone: Europe/Berlin **CloverETL Version:** <unknown>

Preview Import | Import Options << | Commit Import

Import options: [ ] New only

Items to import: [✓] Users [✓] Schedules [✓] Sandboxes [✓] Launch Services [✓] User Groups [✓] Event Listeners [✓] Job Configuration Parameters [✓] Temp Spaces

Select All | Deselect All

Import Log

Summary

Preview of configuration completed without errors. Now you can proceed with actual configuration import by clicking on 'Commit Import'.

Hide Messages << [ ] Show imported items with info-only messages

Configuration does not specify server version.
Preview of server configuration import completed. 1 items would have been added, 1 items updated.

Imported Items

Filter view: [✓ Changes only] All updates

Text View | **Tabular View**

Users & Groups

Users

	User Name	First Name	Last Name	E-mail	Domain	Disabled	Password	Password A1 Digest	Groups
+	johnsmith	John	Smith	smithj@tnet.com	clover	☒	SITx8e1pjoo4em17As3LNw==	70151bae7488da4159ac5ccec97d0995	all_users

User Groups

	Code	Name	Description	Users	Permissions
	all_users	all users		clover, user1, johnsmith	[sandbox_all, user_edit_me, scheduling_all, event_listener_all, run_record_all, launch_config_all, task_log_all, monitoring_all]
		all users		clover, user1	[sandbox_all, user_edit_me, scheduling_all, event_listener_all, run_record_all, launch_config_all, task_log_all, monitoring_all]

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.

Committing 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.

Chapter 18. Diagnostics

CloverETL Server allows you to create a thread dump or a heap dump. The thread and heap dumps are useful for investigation of performance and memory issues.

In server GUI, go to **Configuration** →**System Info** →**Diagnostics**.

Heap Dump

Heap Dump is content of a JVM process memory stored in a binary file.

To download the **Heap Dump**, click **Download** button under **Heap Dump** section. The download of **Heap Dump** may take some time.

The **Dump live objects only** checkbox allows you to avoid dumping of objects awaiting garbage collection

You can use **jvisualvm** or **jhat** to view and analyze the heap dump.

Important

Heap Dump does not work on **WebSphere**. On WebSphere, you can create heap dumps using administration console. See IBM Knowledge Center for the instructions.

Thread Dump

Thread Dump is a list of exisitng JVM threads with their callstacks and held locking objects (if supported). It can be viewed in a text editor.

To download the thread dump, click **Download** button under **Thread Dump** section.

Required Permissions

To create **Thread Dump** or **Heap Dump**, following permissions are required:

- **Configuration** →**Heap Memory Dump**

See also Heap Memory Dump permission (p. 128).

Part V. Using Graphs

Chapter 19. Graph/Jobflow Parameters

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

Keep in mind that `${paramName}` placeholders (parameters) 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 19.1. Defaults for graph execution configuration - see section Graph config properties for details

key	**description**
SANDBOX_CODE	An identifier of a sandbox which contains executed graph.
JOB_FILE	A path to the file (graph, subgraph, jobflow). The path relative to sandbox root path.
SANDBOX_ROOT	An 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. 234) for details.
PARENT_RUN_ID	Run ID of the graph execution which is a 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 a 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 a subgraph, child-job of some jobflow or worker for distributed transformation in cluster. When the execution doesn't have a 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

Graphs executed from a web gui have no additional parameters.

Executed by Launch Service Invocation

Service parameters which have **Pass to graph** attribute 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 with 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 19.2. passed parameters

key	**description**
EVENT_SCHEDULE_EVENT_TYPE	Type of a schedule: SCHEDULE_PERIODIC | SCHEDULE_ONETIME
EVENT_SCHEDULE_LAST_EVENT	Date/time of a 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 incoming message. See details in JMS Messages Listeners (p. 205).

Executed by Task "Start a graph" by Graph/Jobflow Event Listener

Since 3.0 only specified properties from a "source" job are passed to executed job by default. There is a "graph.pass_event_params_to_graph_in_old_style" server config property which can change this behavior so that

ALL parameters from a "source" job are passed to the executed job. This switch is implemented for backwards compatibility. Regarding to the default behaviour: in the editor of graph event listener, you can specify a list of parameters to pass. Please see Start a Graph (p. 170) for details.

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

Table 19.3. passed parameters

key	**description**
EVENT_RUN_SANDBOX	Sandbox with the graph which is source of the event
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 milliseconds which specifies interval of timeout. Makes sense 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 19.4. passed parameters

key	**description**
EVENT_FILE_PATH	Path to the file which is source of the event. Does not contain a file name. Does not end with a file separator. Is passed only for the local file event listener.
EVENT_FILE_NAME	Filename of the file which is source of the event. Is passed only when the "grouping" mode is disabled. Otherwise there are more file events, not just one.
EVENT_FILE_URLS	Contans string, which may be used "as is" in the "file URL" attribute of various CloverETL components. It may contain URL to one or more (if grouping is anabled) files. It may contain local path(s) or remote URL(s) where credentials are replaced by placeholders (due to security reasons).
EVENT_FILE_AUTH_USERNAME	Username/ID to the remote location.
EVENT_FILE_AUTH_USERNAME_URL_ENCODED	The same as EVENT_FILE_AUTH_USERNAME, but the value is also URL encoded, so it may be used in the URL.
EVENT_FILE_AUTH_PASSWORD	Password/key to the remote location. It's encrypted by the master password. It's passed only when the file listener uses user+password authentication.
EVENT_FILE_AUTH_PASSWORD_URL_ENCODED	The same as EVENT_FILE_AUTH_PASSWORD, but the value is also URL encoded, so it may be used in the URL (EVENT_FILE_URLS parameter).

key	description
EVENT_FILE_EVENT_TYPE	SIZE | CHANGE_TIME | APPEARANCE | DISAPPEARANCE
EVENT_FILE_PATTERN	Pattern specified in a 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.

Adding 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. 137).

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 20. Tasks

Task is a graph, jobflow, groovy script, etc. that can be started manually, started on scheduled time, or triggered by some event. Task basically specifies WHAT to do.

There are several tasks implemented for schedule and for graph event listener as follows:

- Start a Graph (p. 170)
- Start a Jobflow (p. 172)
- Abort job (p. 174)
- Execute Shell Command (p. 167)
- Send an Email (p. 163)
- Execute Groovy Code (p. 180)
- Archive Records (p. 175)

Tasks in Cluster Environment

In the Cluster environment, you can specify node where the task runs. The task can run on **Any node** or on **one of selected nodes**. 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.

Tasks are used in

Chapter 22, Scheduling (p. 183)
Chapter 24, Listeners (p. 194)
Chapter 21, Manual Task Execution (p. 182)

Send an Email

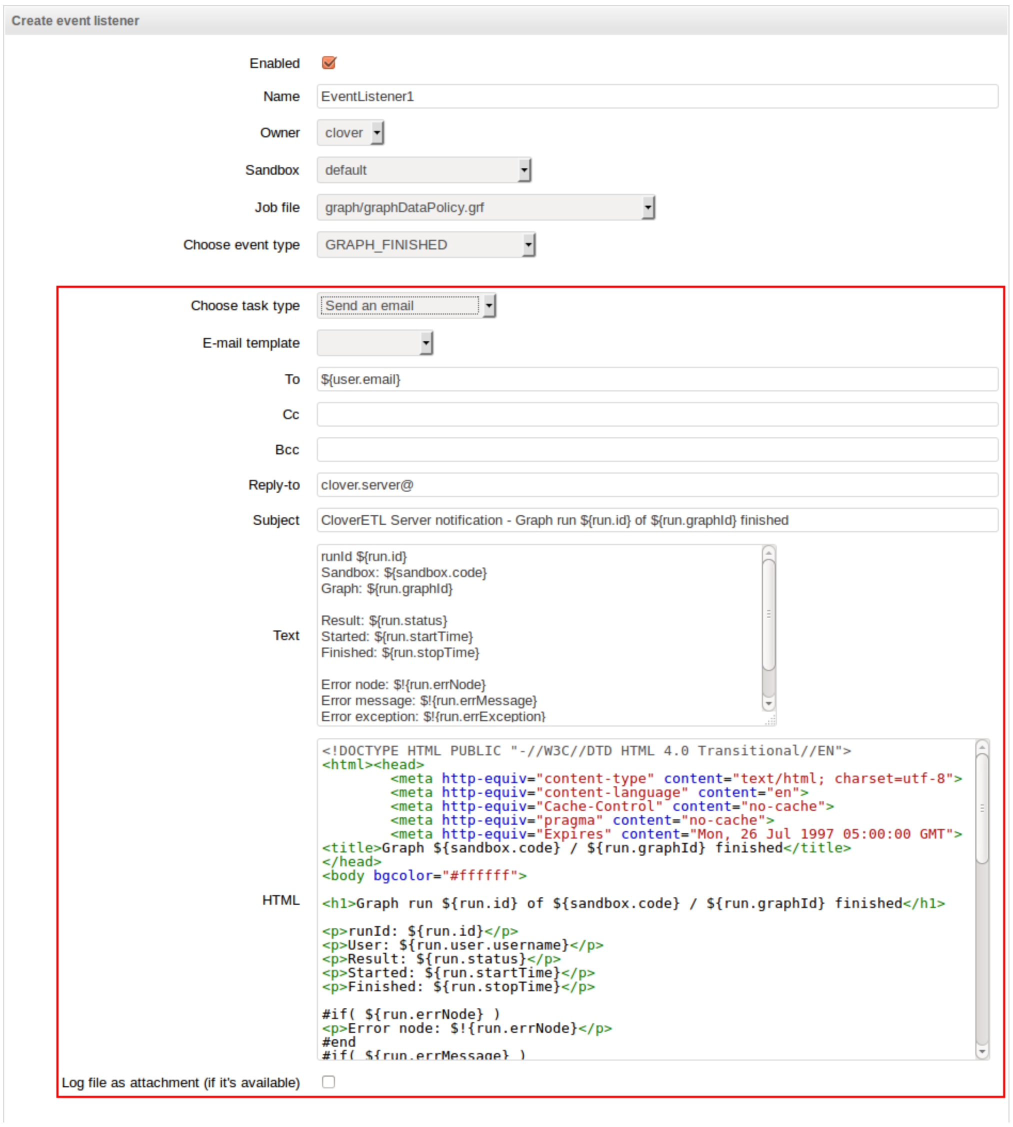
The "send e-mail" task is useful for notifications about result of graph execution. E.g., you can create a listener with this task type to be notified about each failure in specified sandbox or failure of particular graph.

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. 195) for description of this task type).

Note: It seems useless to send e-mails periodically, but it may send current server status or daily summary. These features will be implemented in further versions.

Table 20.1. Attributes of "Send e-mail" task

Task type	"Send an email"
To	Recipient's e-mail address. It is possible to specify more addresses separated by a comma. It is also possible to use placeholders. *See* Placeholders *(p. 164) for details.*
Cc	Cc stands for 'carbon copy'. Copy of the e-mail will be delivered to these addresses. It is possible to specify more addresses separated by a comma. It is also possible to use placeholders. *See* Placeholders *(p. 164) 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 the e-mail.
Reply-to (Sender)	E-mail address of sender. It must be a valid address according to SMTP server. It is also possible to use placeholders. *See* Placeholders *(p. 164) for details.*
Subject	E-mail subject. It is also possible to use placeholders. *See* Placeholders *(p. 164) for details.*
HTML	A body of the e-mail in HTML. The e-mail is created as multipart, so HTML body should have a precedence. A plain text body is only for e-mail clients which do not display HTML. It is also possible to use placeholders. *See* Placeholders *(p. 164) for details.*
Text	A body of the e-mail in plain text. The e-mail is created as multipart, so HTML body should have a precedence. A plain text body is only for e-mail clients which do not display HTML. It is also possible to use placeholders. *See* Placeholders *(p. 164) for details.*
Log file as attachment	If this switch is checked, e-mail will have an attachment with a packed log file of the related graph execution.



Create event listener

Enabled	✓
Name	EventListener1
Owner	clover
Sandbox	default
Job file	graph/graphDataPolicy.grf
Choose event type	GRAPH_FINISHED
Choose task type	Send an email
E-mail template	
To	${user.email}
Cc	
Bcc	
Reply-to	clover.server@
Subject	CloverETL Server notification - Graph run ${run.id} of ${run.graphId} finished

Text

```
runId ${run.id}
Sandbox: ${sandbox.code}
Graph: ${run.graphId}

Result: ${run.status}
Started: ${run.startTime}
Finished: ${run.stopTime}

Error node: $!{run.errNode}
Error message: $!{run.errMessage}
Error exception: $!{run.errException}
```

HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <meta http-equiv="content-language" content="en">
        <meta http-equiv="Cache-Control" content="no-cache">
        <meta http-equiv="pragma" content="no-cache">
        <meta http-equiv="Expires" content="Mon, 26 Jul 1997 05:00:00 GMT">
<title>Graph ${sandbox.code} / ${run.graphId} finished</title>
</head>
<body bgcolor="#ffffff">

<h1>Graph run ${run.id} of ${sandbox.code} / ${run.graphId} finished</h1>

<p>runId: ${run.id}</p>
<p>User: ${run.user.username}</p>
<p>Result: ${run.status}</p>
<p>Started: ${run.startTime}</p>
<p>Finished: ${run.stopTime}</p>

#if( ${run.errNode} )
<p>Error node: $!{run.errNode}</p>
#end
#if( ${run.errMessage} )
```

Log file as attachment (if it's available) ☐

Figure 20.1. Web GUI - send e-mail

Note: Do not forget to configure connection to SMTP server (See Part III, "Configuration" (p. 60) for details).

Placeholders

Placeholder may be used in some fields of tasks. They are especially useful for e-mail tasks, where you can generate content of e-mail according to context variables.

Note: In most cases, you can avoid this by using e-mail templates (See E-mail 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 depending on type of event. E.g., if task is processed because of graph event, then *run* and *sandbox* variables contain related data, otherwise they are empty.

Table 20.2. Placeholders useful in e-mail templates

Variable name	Contains
now	Current date-time
user	User, who caused this event. It may be an owner of a schedule, or someone who executed a graph. It contains sub-properties, which are accessible using dot notation (i.e. ${user.email}) e-mail: • user.email • user.username • user.firstName • user.lastName • user.groups (list of values)
run	Data structure describing one single graph execution. It contains sub-properties, which are accessible using dot notation (i.e. ${run.jobFile}) • job.jobFile • job.status • job.startTime • job.stopTime • job.errNode • job.errMessage • job.errException • job.logLocation
tracking	A data structure describing status of components in graph execution. It contains sub-properties, which are accessible using Velocity syntax for loops and conditions. `#if (${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></tr>` `   #foreach ($node in $phase.trackingNodes)` `      <tr><td>${node.nodeName}</td>` `          <td>${node.result}</td>` `          <td></td><td></td><td></td></tr>` `      #foreach ($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 a sandbox containing the executed graph. It contains sub-properties, which are accessible using dot notation (i.e. ${sandbox.name}) • sandbox.name • sandbox.code • sandbox.rootPath
schedule	Data structure describing schedule which triggered this task. It contains sub-properties, which are accessible using dot notation (i.e. ${schedule.description}) • schedule.description • schedule.startTime • schedule.endTime • schedule.lastEvent • schedule.nextEvent • schedule.fireMisfired

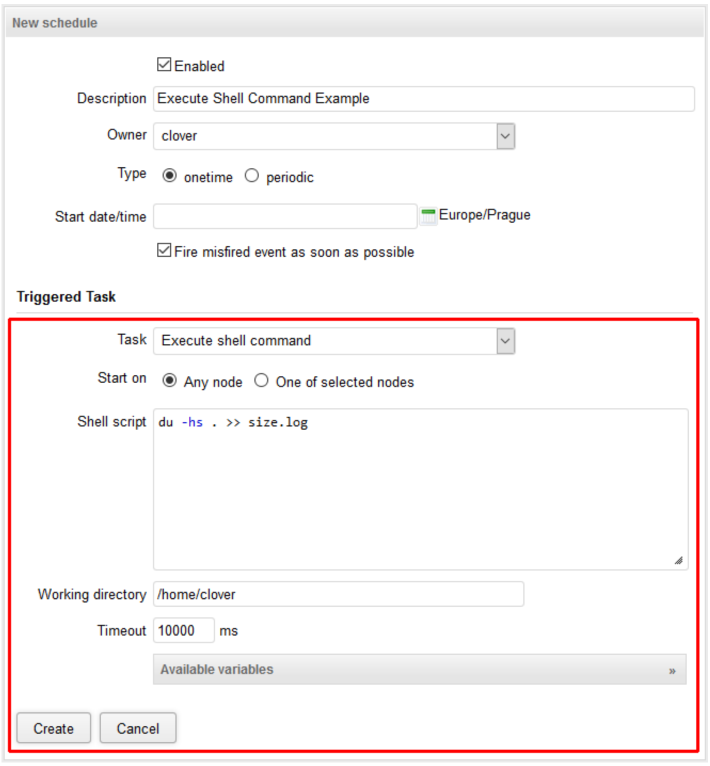
Execute Shell Command

Execute Shell Command executes system command or a shell script.

This task is used is used in Chapter 22, Scheduling (p. 183) Chapter 24, Listeners (p. 194) and Chapter 21, Manual Task Execution (p. 182).

Table 20.3. Attributes of "Execute shell command" task

Task type	"Execute shell command"
Start on	Node IDs to process the task This attribute is accessible only in the cluster environment. If there are nodes specified, the task will be processed on the first node which is online and ready.
Shell script	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.


New schedule

Enabled: ✓
Description: Execute Shell Command Example
Owner: clover
Type: [✓ onetime] periodic
Start date/time: Europe/Prague
Fire misfired event as soon as possible: ✓

Triggered Task

Task: Execute shell command
Start on: [✓ Any node] One of selected nodes
Shell script:
```
du -hs . >> size.log
```
Working directory: /home/clover
Timeout: 10000 ms

Figure 20.2. Web GUI - shell command

Execute Shell Command Parameters

Some parameters are available only in particular context: scheduling, event listeners, or manual task execution.

Table 20.4. Parameters of "Execute shell command" task

event	Event that has triggered the task
now	Current date-time
task	The triggered task
user	Object representing user who executed the graph/jobflow. It contains sub-properties that are accessible using dot notation (i.e. ${user.email}) • user.email • user.username • user.firstName • user.lastName • user.groups (list of values)

Table 20.5. Parameters of "Execute shell command" task - available in scheduling

schedule	Object representing the schedule that triggered this task. It contains sub-properties that are accessible using dot notation (i.e. ${schedule.description}) • schedule.description • schedule.startTime • schedule.endTime • schedule.lastEvent • schedule.nextEvent • schedule.fireMisfired
EVENT_USERNAME	Username of the user who caused the event.
EVENT_USER_ID	Numeric ID of the user who caused the event.
EVENT_SCHEDULE_DESCRIPTION	Description of the schedule.
EVENT_SCHEDULE_EVENT_TYPE	Type of the schedule - SCHEDULE_ONETIME or SCHEDULE_PERIODIC.
EVENT_SCHEDULE_ID	Numeric ID of the schedule
EVENT_SCHEDULE_LAST_EVENT	Date-time of the latest schedule triggering (in java.util.Date.toString() format).

Table 20.6. Parameters of "Execute shell command" task - available in listeners

run	Object representing single graph/jobflow execution. It contains sub-properties that are accessible using dot notation (i.e. ${run.jobFile}). • run.jobFile • run.status • run.startTime • run.stopTime • run.errNode • run.errMessage • run.errException

sandbox	Object representing a sandbox containing the executed graph/jobflow. It contains sub-properties that are accessible using dot notation (i.e. ${sandbox.name}) • sandbox.name • sandbox.code • sandbox.rootPath
tracking	Object represeting status of components in a graph execution. It contains sub-properties that are accessible using Velocity syntax for loops and conditions.
EVENT_USERNAME	Username of the user who caused the event.
EVENT_USER_ID	Numeric ID of the user who caused the event.
EVENT_RUN_SANDBOX	Code of the sandbox containing the graph/jobflow.
EVENT_RUN_JOB_FILE	Sandbox-relative path to the graph/jobflow file.
EVENT_RUN_RESULT	Current status of the graph/jobflow execution • N_A • READY • RUNNING • WAITING • FINISHED_OK • ERROR • ABORTED • TIMEOUT • UNKNOWN
EVENT_RUN_ID	Numeric ID of the run record representing graph/jobflow execution
EVENT_TIMEOUT	Specified timeout (in milliseconds) for the TIMEOUT event to occur.
EVENT_JOB_EVENT_TYPE	Graph event that triggered the task • GRAPH_STARTED • GRAPH_PHASE_FINISHED • GRAPH_FINISHED • GRAPH_ERROR • GRAPH_ABORTED • GRAPH_TIMEOUT • GRAPH_STATUS_UNKNOWN

Table 20.7. Parameters of "Execute shell command" task - available in manual task execution

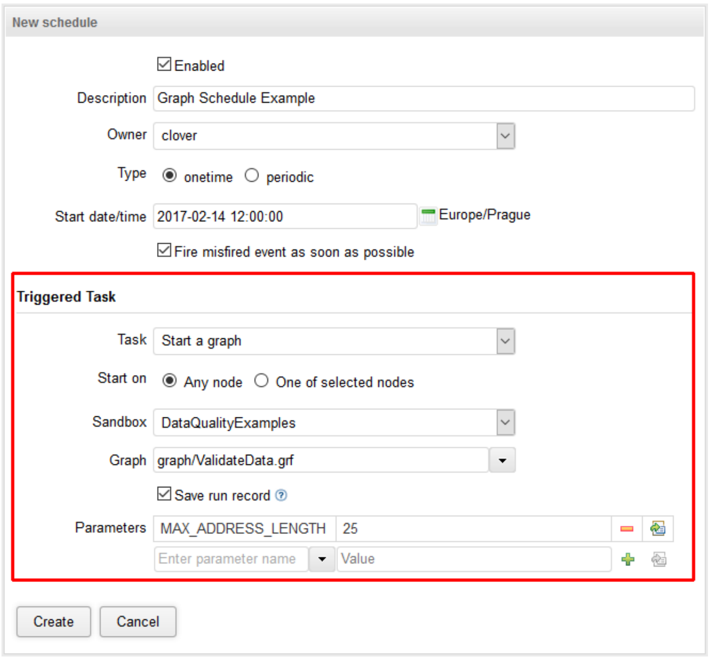
parameters	Task parameters - container for String-String key-value pairs passed to this task.

Start a Graph

Start a Graph starts a specified graph from specified sandbox.

Table 20.8. Attributes of "Graph execution" task

Task type	"Start a graph"
Start on	Node(s) to process the task.
Sandbox	This select box contains sandboxes which are readable for logger user. Select sandbox which contains graph to execute.
Graph	The graph to be executed. This select box is filled with all graphs files accessible in selected sandbox. Type a graph name or path to filter available items.
Save run record	Saves run record to database. If the task runs too often (once in several seconds), you can increase the database performance by disabling this.
Parameters	List of parameters passed to the graph. Event parameters like "EVENT_RUN_RESULT", "EVENT_RUN_ID" etc. are passed to the executed job without limitations. Parameters EVENT_RUN_RESULT and EVENT_RUN_ID are used in context of event listeners. They are not used in context of scheduling.


New schedule

☑ Enabled

Description: Graph Schedule Example

Owner: clover

Type: [✓ onetime] periodic

Start date/time: 2017-02-14 12:00:00 Europe/Prague

☑ Fire misfired event as soon as possible

Triggered Task

Task: Start a graph

Start on: [✓ Any node] One of selected nodes

Sandbox: DataQualityExamples

Graph: graph/ValidateData.grf

☑ Save run record

Parameters: MAX_ADDRESS_LENGTH 25

Enter parameter name Value

Figure 20.3. Web GUI - Graph execution task

Please note that behaviour of this task type is almost the same as Start a Jobflow (p. 172).

Parameters

You can start a graph with parameters.

To start a graph with parameter choose an existing parameter from a list, set its value, and click the plus sign button at the end of line.

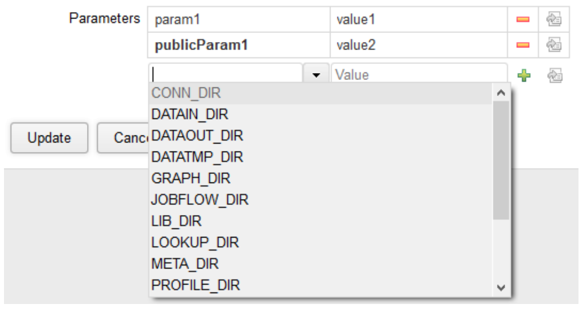


Figure 20.4. Web GUI - Graph execution task

If the graph is started by event listener, it receives some additional parameters from the triggering job.

Parameters passed to graph by Event Listeners

Table 20.9. Additional parameters available in Event Listeners

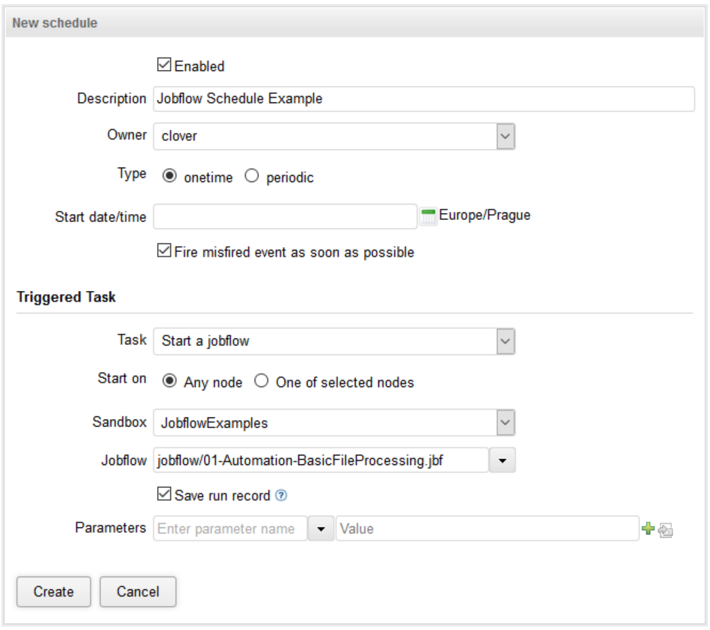
EVENT_USERNAME	Username of the user who caused the event
EVENT_USER_ID	Numeric ID of the user who caused the event.
EVENT_RUN_SANDBOX	Code of the sandbox containing the graph/jobflow
EVENT_RUN_JOB_FILE	Sandbox-relative path to the graph/jobflow file.
EVENT_RUN_RESULT	Current status of the graph/jobflow execution (N_A, READY, RUNNING, WAITING, FINISHED_OK, ERROR, ABORTED, TIMEOUT or UNKNOWN).
EVENT_RUN_ID	Numeric ID of the run record representing graph/jobflow execution.
EVENT_JOB_EVENT_TYPE	Graph/Jobflow event type that triggered the task.
EVENT_TIMEOUT	Specified timeout (in milliseconds) for the TIMEOUT event to occur

Start a Jobflow

Start a jobflow starts a specified jobflow from a specified sandbox.

Table 20.10. Attributes of "Jobflow execution" task

Task type	"Start a jobflow"
Start on	Node(s) to process the task This attribute is accessible only in the cluster environment. 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.
Save run record	Saves run record to database. If the task runs too often (once in several seconds), you can increase the database performance by disabling this.
Parameters	Key-value pairs which are passed to the executed job as parameters. Event parameters like "EVENT_RUN_RESULT", "EVENT_RUN_ID" etc. are passed to the executed job without limitations. Parameters EVENT_RUN_RESULT and EVENT_RUN_ID are used in context of event listeners. They are not used in context of scheduling.


New schedule

☑ Enabled

Description: Jobflow Schedule Example

Owner: clover

Type: [✓ onetime] periodic

Start date/time: Europe/Prague

☑ Fire misfired event as soon as possible

Triggered Task

Task: Start a jobflow

Start on: [✓ Any node] One of selected nodes

Sandbox: JobflowExamples

Jobflow: jobflow/01-Automation-BasicFileProcessing.jbf

☑ Save run record

Parameters: Enter parameter name | Value

Figure 20.5. Web GUI - Jobflow execution task

Please note that behaviour of this task type is almost the same as Start a Graph (p. 170).

If the jobflow start is triggered by an event, the same set of parameters as in graph event listener is passed to jobflow. Parameters (p. 171).

Start a Profiler Job

Start a profiler job starts a specified profiler job from a specified directory.

You can pass parameters to the profiler job in the same way as in case you Start a Graph (p. 170) or Start a Jobflow (p. 172).

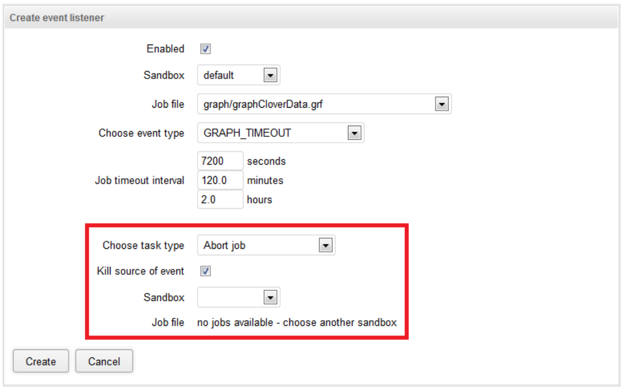
In case of triggering the profiler job by event listener, the same set of additional parameters, as in case of execution of graph, is passed to the profiler job. See Parameters passed to graph by Event Listeners (p. 171).

Abort job

This task kills/aborts specified job (ETL graph or jobflow), if it is currently running.

Table 20.11. Attributes of "Abort job" task

Task type	"Abort job"
Start on	Node(s) to process the task. This attribute is accessible only in the cluster environment. 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 the job which is source of the event, which activated this task. Attributes sandbox and job are ignored. This checkbox makes sense only if **Abort job** is activated by some event.
Sandbox	Select a sandbox which contains the 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 that are currently running and will be killed. This attribute works only when "Kill source of event" switch is off.


Create event listener

Enabled	✓
Sandbox	default
Job file	graph/graphCloverData.grf
Choose event type	GRAPH_TIMEOUT
Job timeout interval	7200 seconds 120.0 minutes 2.0 hours
Choose task type	Abort job
Kill source of event	✓
Sandbox	
Job file	no jobs available - choose another sandbox

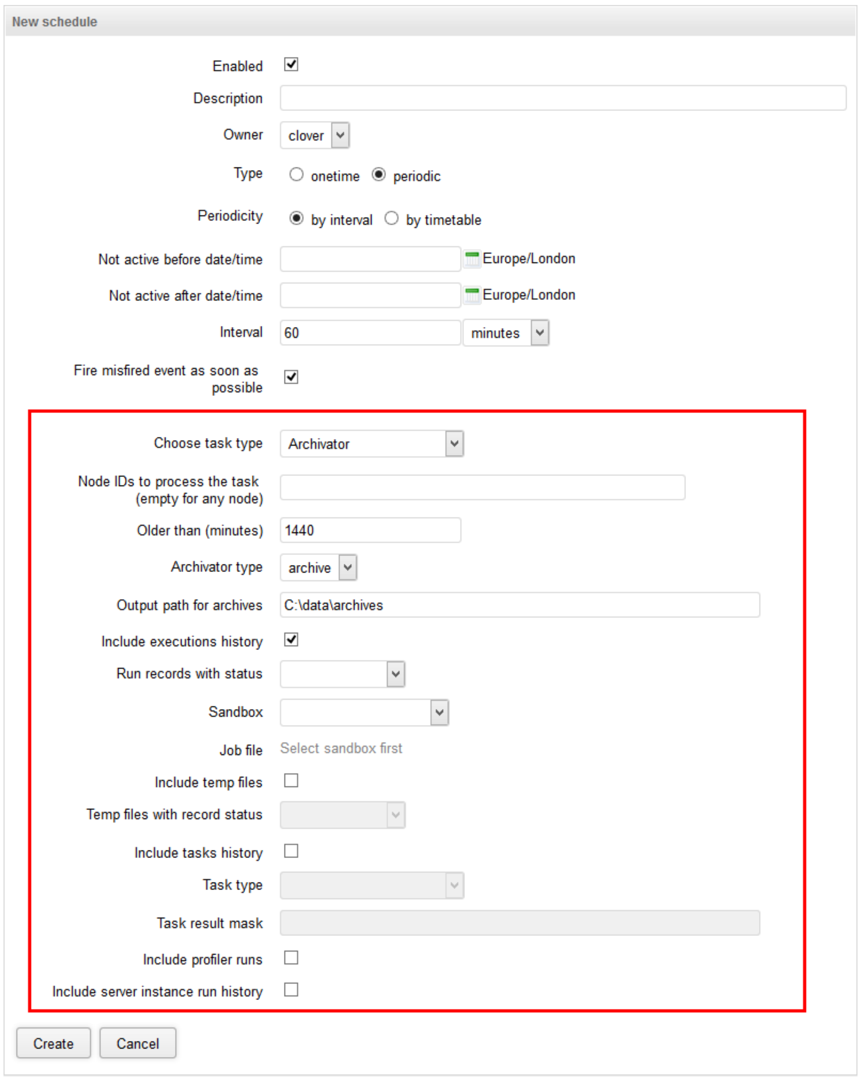
Figure 20.6. Web GUI - "Abort job"

Archive Records

This task can archive (or delete) obsolete records from DB.

Table 20.12. Attributes of "Archivator" task

Task type	"Archivator"
Start on	This attribute specifies cluster node on which may process the task. This attribute is accessible only in the cluster environment. 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.
Archivator type	There are two possible values: "archive" or "delete". Delete option removes records without any possibility of UNDO operation. Archive option removes records from DB, but creates a ZIP package with CSV files containing deleted data.
Older then	Time period (in minutes) - it specifies which records are evaluated as obsolete. Records older then the specified interval are stored in archives.
Output path for archives	This attribute makes sense only for "archive" type.
Include executions history	
Include temp files	If checked, archivator removes all graph temporary files older than the given timestamp defined in "Older than" attribute. The temporary files are files with graph debug data, dictionary files and files created by graph components.
Include tasks history	If checked, archivator will include run records. Log files of graph runs are included as well.
Include profiler runs	If checked, archivator will include profiler job results.
Include server instance history	



New schedule

Enabled	✓
Description	
Owner	clover
Type	onetime [✓ periodic]
Periodicity	[✓ by interval] by timetable
Not active before date/time	Europe/London
Not active after date/time	Europe/London
Interval	60 minutes
Fire misfired event as soon as possible	✓
Choose task type	Archivator
Node IDs to process the task (empty for any node)	
Older than (minutes)	1440
Archivator type	archive
Output path for archives	C:\data\archives
Include executions history	✓
Run records with status	
Sandbox	
Job file	Select sandbox first
Include temp files	☐
Temp files with record status	
Include tasks history	☐
Task type	
Task result mask	
Include profiler runs	☐
Include server instance run history	☐

Figure 20.7. Web GUI - archive records

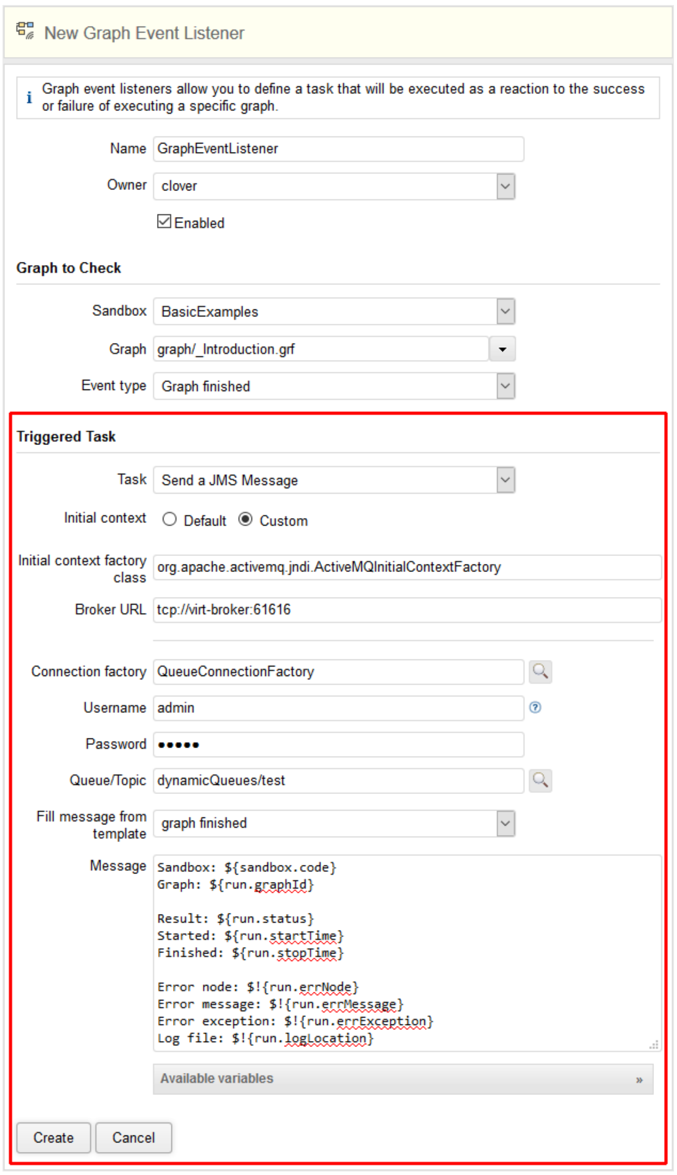
Send a JMS Message

This type of task is useful for notifications about result of graph execution. E.g. you can create a graph event listener with this task type to be notified about each failure in a specific sandbox or failure of a 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 20.13. Attributes of JMS message task

Task type	"JMS message"
Initial context	Choose between **default** and **custom** initial context.
Initial context class name	A full class name of javax.naming.InitialContext implementation. Each JMS provider has its own implementation. E.g., for Apache MQ it is "org.apache.activemq.jndi.ActiveMQInitialContextFactory". If it is empty, server uses the default initial context.
Broker URL	
Connection factory JNDI name	A JNDI name of a connection factory. It depends on a JMS provider.
Destination JNDI name	JNDI name of a message queue/topic on the server
Username	Username for connection to a JMS message broker
Password	Password for connection to a JMS message broker
URL	URL of a JMS message broker
JMS pattern	This select box is available only when user is creating a 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 a JMS message. It is also possible to use placeholders. *See* Placeholders *(p. 164) of send e-mail task for details.*



New Graph Event Listener

Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name: GraphEventListener
Owner: clover
☑ Enabled

Graph to Check

Sandbox: BasicExamples
Graph: graph/_Introduction.grf
Event type: Graph finished

Triggered Task

Task: Send a JMS Message
Initial context: Default [✓ Custom]
Initial context factory class: org.apache.activemq.jndi.ActiveMQInitialContextFactory
Broker URL: tcp://virt-broker:61616

Connection factory: QueueConnectionFactory
Username: admin
Password: •••••
Queue/Topic: dynamicQueues/test
Fill message from template: graph finished
Message:

```
Sandbox: ${sandbox.code}
Graph: ${run.graphId}

Result: ${run.status}
Started: ${run.startTime}
Finished: ${run.stopTime}

Error node: $!{run.errNode}
Error message: $!{run.errMessage}
Error exception: $!{run.errException}
Log file: $!{run.logLocation}
```

Figure 20.8. Web GUI - Task JMS message editor

Table 20.14. Parameters of "Send a JMS Message"

event	Event that has triggered the task.
now	Current date-time
task	The triggered task.
user	Object representing owner of the schedule. It contains sub-properties that are accessible using dot notation (i.e. ${user.email}) e-mail, username, firstName, lastName, groups (list of values).
schedule	Object representing the schedule that triggered this task. It contains sub-properties that are accessible using dot notation (i.e. ${schedule.description}) description, startTime, endTime, lastEvent, nextEvent, fireMisfired.
EVENT_USERNAME	Username of the user who caused the event
EVENT_USER_ID	Numeric ID of the user who caused the event.
EVENT_SCHEDULE_DESCRIPTION	Description of the schedule

EVENT_SCHEDULE_EVENT_TYPE	Type of the schedule - SCHEDULE_ONETIME or SCHEDULE_PERIODIC.
EVENT_SCHEDULE_ID	Numeric ID of the schedule.
EVENT_SCHEDULE_LAST_EVENT	Date-time of the latest schedule triggering (in java.util.Date.toString() format).

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.

If source codes are provided from both a file and through the input form only the code from the input form will be executed.

In cluster environment there is also one additional attribute "Node IDs to 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 20.15. 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 a 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
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 only available for ScheduleServerEvent as event variable class.

Example of use Groovy script

This example shows a script which writes text file describing the 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();
```

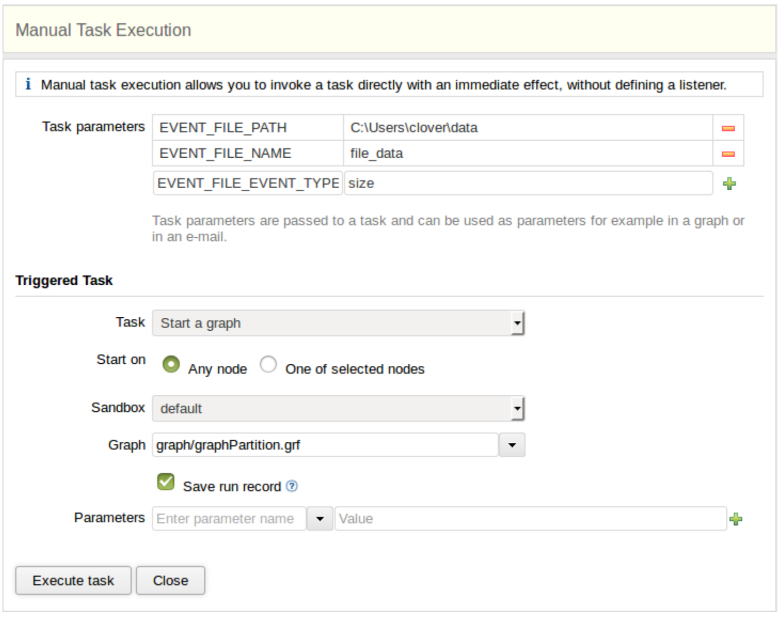
Chapter 21. Manual Task Execution

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".


Manual Task Execution

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

Task parameters

EVENT_FILE_PATH	C:\Users\clover\data
EVENT_FILE_NAME	file_data
EVENT_FILE_EVENT_TYPE	size

Task parameters are passed to a task and can be used as parameters for example in a graph or in an e-mail.

Triggered Task

Task: Start a graph

Start on: [✓ Any node] One of selected nodes

Sandbox: default

Graph: graph/graphPartition.grf

[✓] Save run record

Parameters: Enter parameter name | Value

Figure 21.1. Web GUI - "Manual task execution" form

Using Manual Task Execution

In server GUI, switch to **Event Listeners** tab. In the **New Listener** drop-down menu, select the **Manual Task Execution** option.

Choose the task type you would like to use. See also documenation on chosen tasks:

Send an Email (p. 163)
Execute Shell Command (p. 167)
Start a Graph (p. 170)
Start a Jobflow (p. 172)
Start a Profiler Job (p. 173)
Abort job (p. 174)
Archive Records (p. 175)
Send a JMS Message (p. 177)
Execute Groovy Code (p. 180)

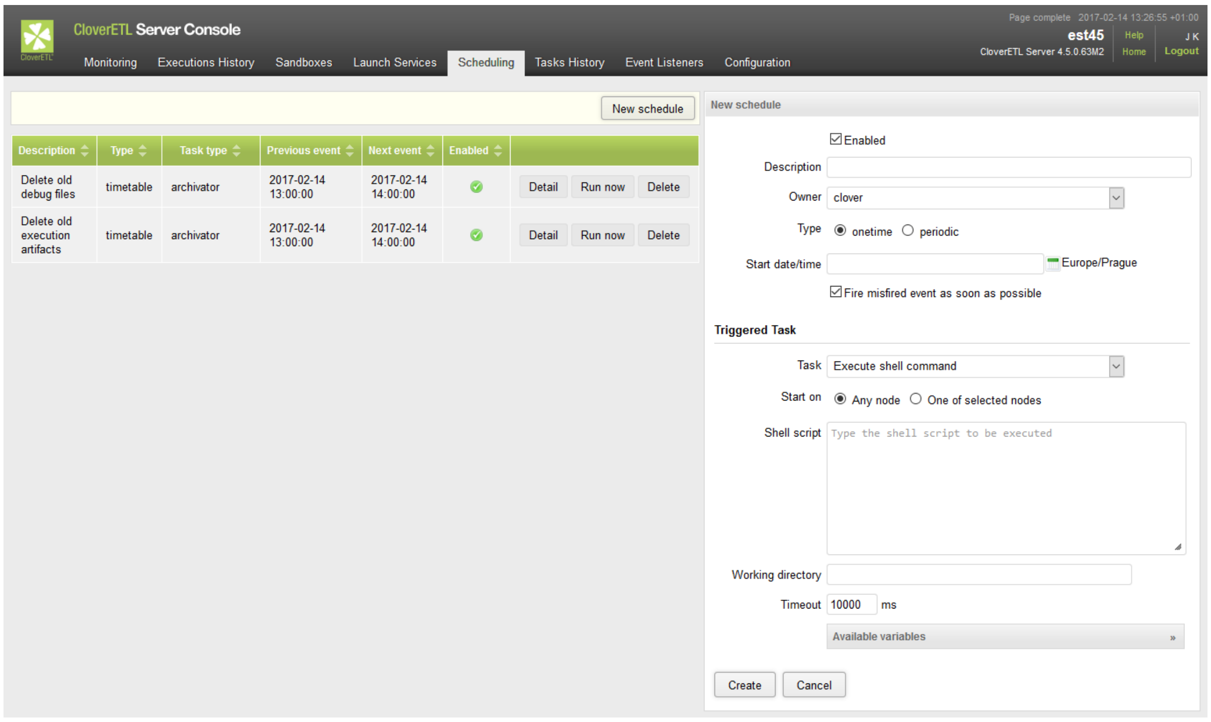
To access the **Manual Task Execution** form, you need Manual task execution permission (p. 123).

Chapter 22. 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).



Description	Type	Task type	Previous event	Next event	Enabled	
Delete old debug files	timetable	archivator	2017-02-14 13:00:00	2017-02-14 14:00:00	✓	Detail Run now Delete
Delete old execution artifacts	timetable	archivator	2017-02-14 13:00:00	2017-02-14 14:00:00	✓	Detail Run now Delete

New schedule

[✓ Enabled]

Description	
Owner	clover
Type	[✓ onetime] periodic
Start date/time	Europe/Prague
	[✓ Fire misfired event as soon as possible]

Triggered Task

Task	Execute shell command
Start on	[✓ Any node] One of selected nodes
Shell script	Type the shell script to be executed
Working directory	
Timeout	10000 ms

Figure 22.1. Web GUI - section "Scheduling" - create new

The tasks you can schedule are described in Chapter 20, Tasks (p. 162).

Send an Email (p. 163)
Execute Shell Command (p. 167)
Start a Graph (p. 170)
Start a Jobflow (p. 172)
Start a Profiler Job (p. 173)
Abort job (p. 174)
Archive Records (p. 175)
Send a JMS Message (p. 177)
Execute Groovy Code (p. 180)

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.

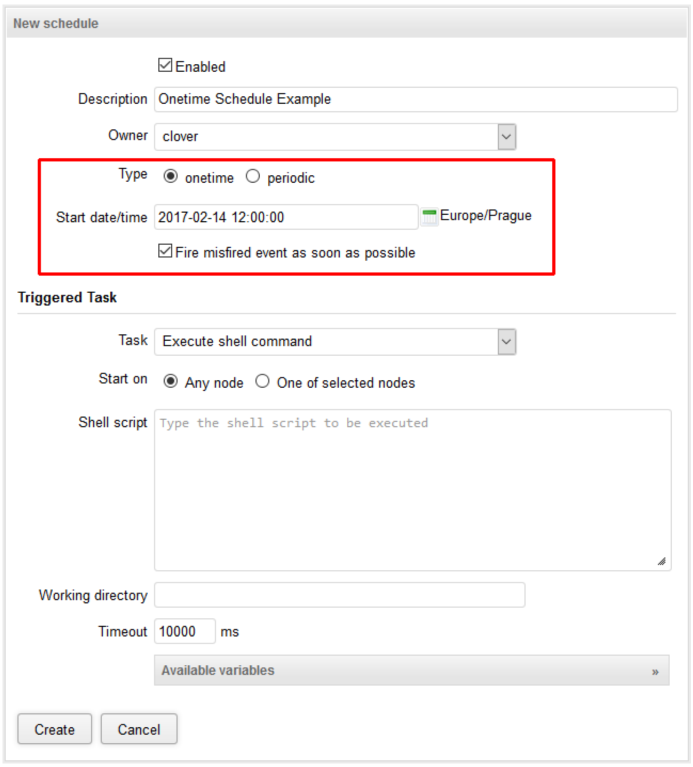
- Onetime Schedule (p. 184)
- Periodical schedule by Interval (p. 186)
- Periodical schedule by timetable (Cron Expression) (p. 187)

Onetime Schedule

This schedule is triggered just once.

Table 22.1. Onetime schedule attributes

Type	"onetime"
Start date/time	Date and time, specified with minutes precision.
Fire misfired event as soon as possible	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.


New schedule

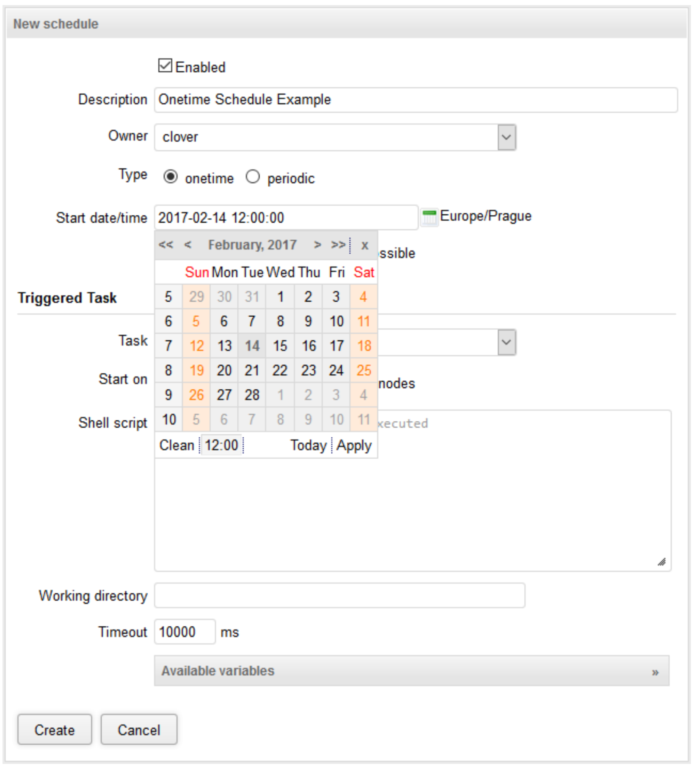
☑ Enabled

Description	Onetime Schedule Example
Owner	clover
Type	[✓ onetime] periodic
Start date/time	2017-02-14 12:00:00 Europe/Prague
	☑ Fire misfired event as soon as possible

Triggered Task

Task	Execute shell command
Start on	[✓ Any node] One of selected nodes
Shell script	Type the shell script to be executed
Working directory	
Timeout	10000 ms

Figure 22.2. Web GUI - onetime schedule form



New schedule

[✓] Enabled

Description	Onetime Schedule Example
Owner	clover
Type	[✓ onetime] periodic
Start date/time	2017-02-14 12:00:00 Europe/Prague

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
5	29	30	31	1	2	3	4
6	5	6	7	8	9	10	11
7	12	13	14	15	16	17	18
8	19	20	21	22	23	24	25
9	26	27	28	1	2	3	4
10	5	6	7	8	9	10	11

Working directory	
Timeout	10000 ms

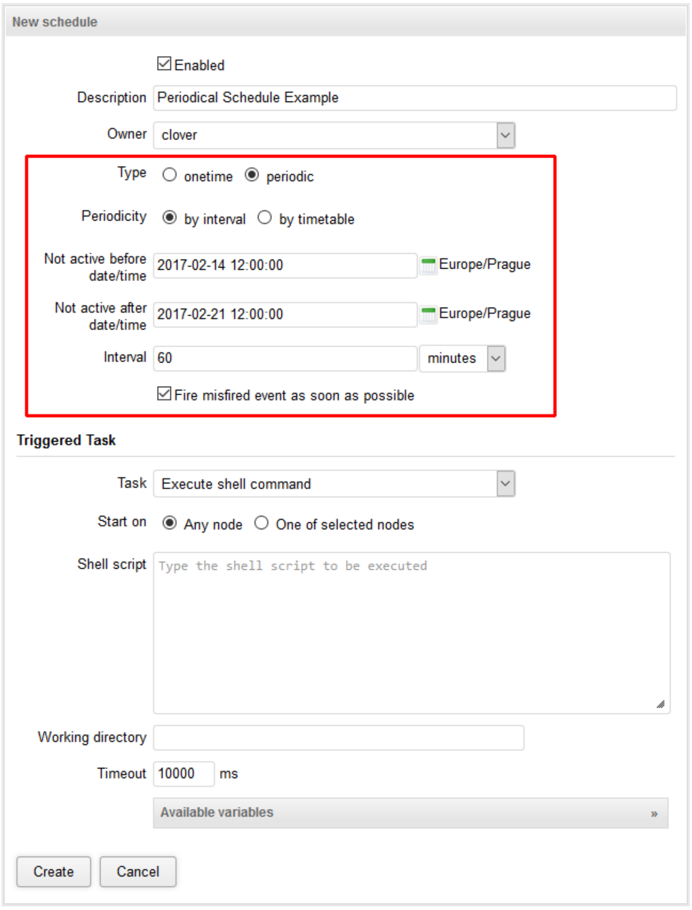
Figure 22.3. Web GUI - schedule form - calendar

Periodical schedule by Interval

This type of schedule is the simplest periodical type. Trigger times are specified by these attributes:

Table 22.2. 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 event as soon as possible	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.


New schedule

	☑ Enabled
Description	Periodical Schedule Example
Owner	clover
Type	○ onetime [✓ ● periodic]
Periodicity	[✓ ● by interval] ○ by timetable
Not active before date/time	2017-02-14 12:00:00 Europe/Prague
Not active after date/time	2017-02-21 12:00:00 Europe/Prague
Interval	60 minutes
	☑ Fire misfired event as soon as possible

Triggered Task

Task	Execute shell command
Start on	[✓ ● Any node] ○ One of selected nodes
Shell script	Type the shell script to be executed
Working directory	
Timeout	10000 ms

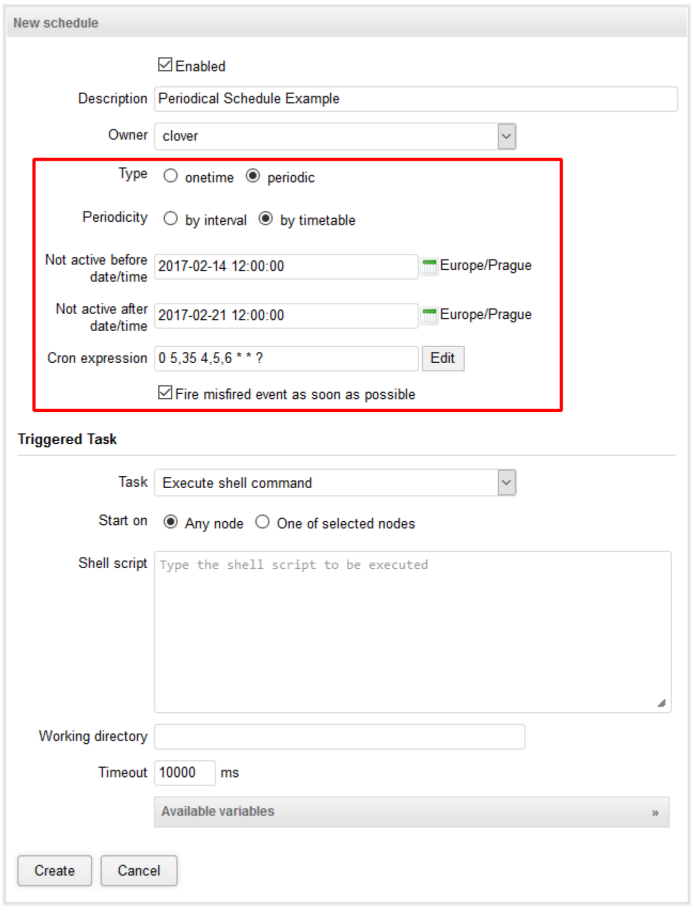
Figure 22.4. Web GUI - periodical schedule form

Periodical schedule by timetable (Cron Expression)

Timetable is specified by powerful (but a little bit tricky) cron expression.

Table 22.3. Cron periodical schedule attributes

Type	"periodic"
Periodicity	"by timetable"
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 event as soon as possible	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.


New schedule

[✓] Enabled

Description	Periodical Schedule Example
Owner	clover
Type	onetime [✓ periodic]
Periodicity	by interval [✓ by timetable]
Not active before date/time	2017-02-14 12:00:00 Europe/Prague
Not active after date/time	2017-02-21 12:00:00 Europe/Prague
Cron expression	0 5,35 4,5,6 * * ?

[✓] Fire misfired event as soon as possible

Triggered Task

Task	Execute shell command
Start on	[✓ Any node] One of selected nodes
Shell script	Type the shell script to be executed
Working directory	
Timeout	10000 ms

Figure 22.5. Cron periodical schedule form

The **Edit** button helps you to set up cron expression even without knowledge of the exact cron expression format.

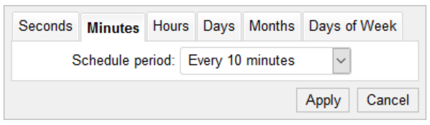
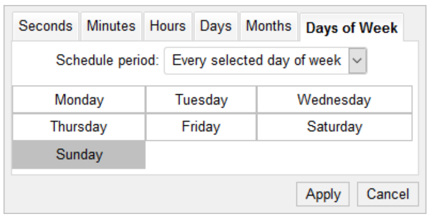

Schedule period: Every 10 minutes

Figure 22.6. Editing the cron expression - minutes



Schedule period: Every selected day of week

Monday	Tuesday	Wednesday
Thursday	Friday	Saturday
Sunday		

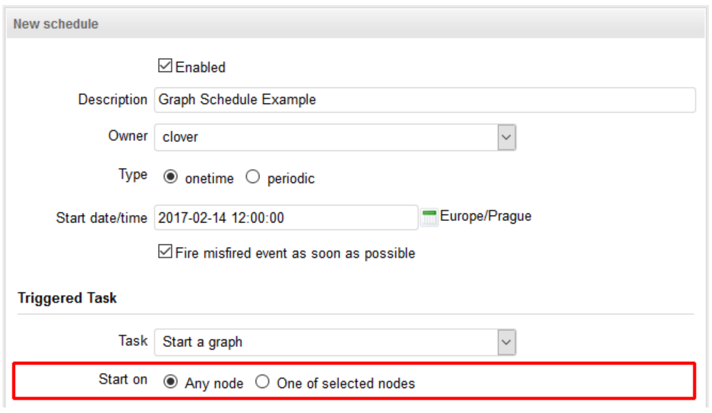
Figure 22.7. Editing the cron expression - days of week

Allocations of Scheduled Task on Nodes

In cluster environment, you can set node on which the scheduled task will be launched.

The task can be scheduled on arbitrary node or on one or more specified nodes.

Allocation on Any Arbitrary Node



New schedule

[✓] Enabled

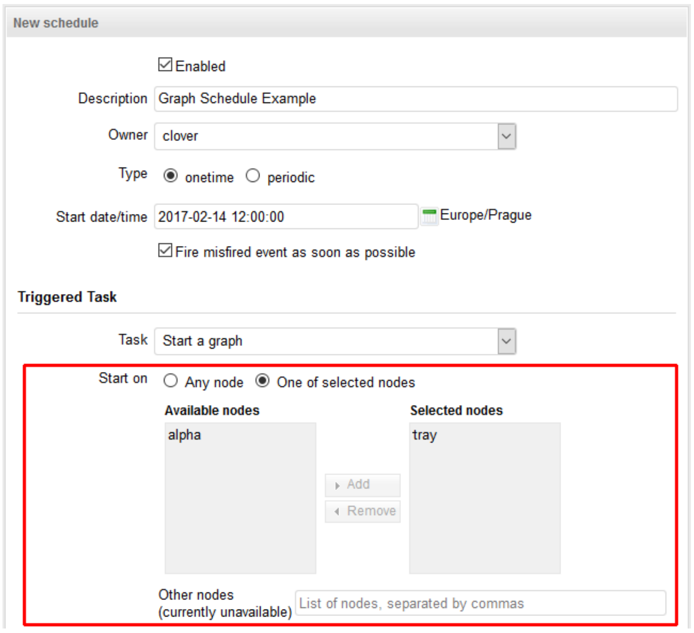
Description	Graph Schedule Example
Owner	clover
Type	[✓ onetime] periodic
Start date/time	2017-02-14 12:00:00 Europe/Prague
	[✓] Fire misfired event as soon as possible

Triggered Task

Task	Start a graph
Start on	[✓ Any node] One of selected nodes

Figure 22.8. Schedule allocation - Any node

Allocation on One or More Chosen Nodes



New schedule

[✓] Enabled

Description	Graph Schedule Example
Owner	clover
Type	[✓ onetime] periodic
Start date/time	2017-02-14 12:00:00 Europe/Prague
	[✓] Fire misfired event as soon as possible

Triggered Task

Task	Start a graph
Start on	Any node [✓ One of selected nodes]

Available nodes		Selected nodes
alpha	Add / Remove	tray

Other nodes (currently unavailable): List of nodes, separated by commas

Figure 22.9. Schedule allocation - One ore more specific nodes

Scheduling the Tasks - Examples

Start a graph at specific time

This example shows scheduling the start of a graph at the specific time.

1. In server GUI, in **Scheluling** section, click **New schedule**.
2. Enter the description that allows you to successfully identify the task between other existing tasks, e.g. **My one-time processing example**.
3. Choose **onetime** to start the graph just once.
4. Enter start date and time. The calendar accessible under calendar icon might help you to enter the required date and time in correct format.
5. Choose **Task Start a graph**.
6. Select the **sandbox** and **graph** within the sandbox.

Start a Jobflow once an hour

This example show scheduling the repetitive run.

Create a new schedule that runs **UserStats.jbf** jobflow from **reports** sandbox once an hour.

1. Enter **description**, e.g. **Hourly user stats**.
2. Choose **periodic type** and **periodicity by interval**.
3. The task will be started once an hour within some time period. Enter the beginning of this period (**not active before**) and end of the period (**not active after**).
4. Enter **interval** between two jobflow starts.
5. Select the **sandbox** and **graph** within the sandbox.

Complex Scheduling

This example shows complex scheduling using cron expression.

Start a graph **WeekendGraph.grf** every Saturday and Sunday at 22:15.

1. Enter the description of the schedule, e.g. **The Weekend Graph**
2. Select **periodic type** and **periodicity by timetable**.
3. Click **Edit** on line starting with **Cron expression**.

   On **Minutes** tab, select **Every selected minute** and select **15**.

   On **Hours** tab, select **Every selected hour** and select **22**.

   On **Days of Week** tab, select **Every selected day of week** and click **Saturday** and **Sunday**.
4. The **Task** is **Start a graph**
5. Select the **sandbox** and **graph**.

Chapter 23. 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.

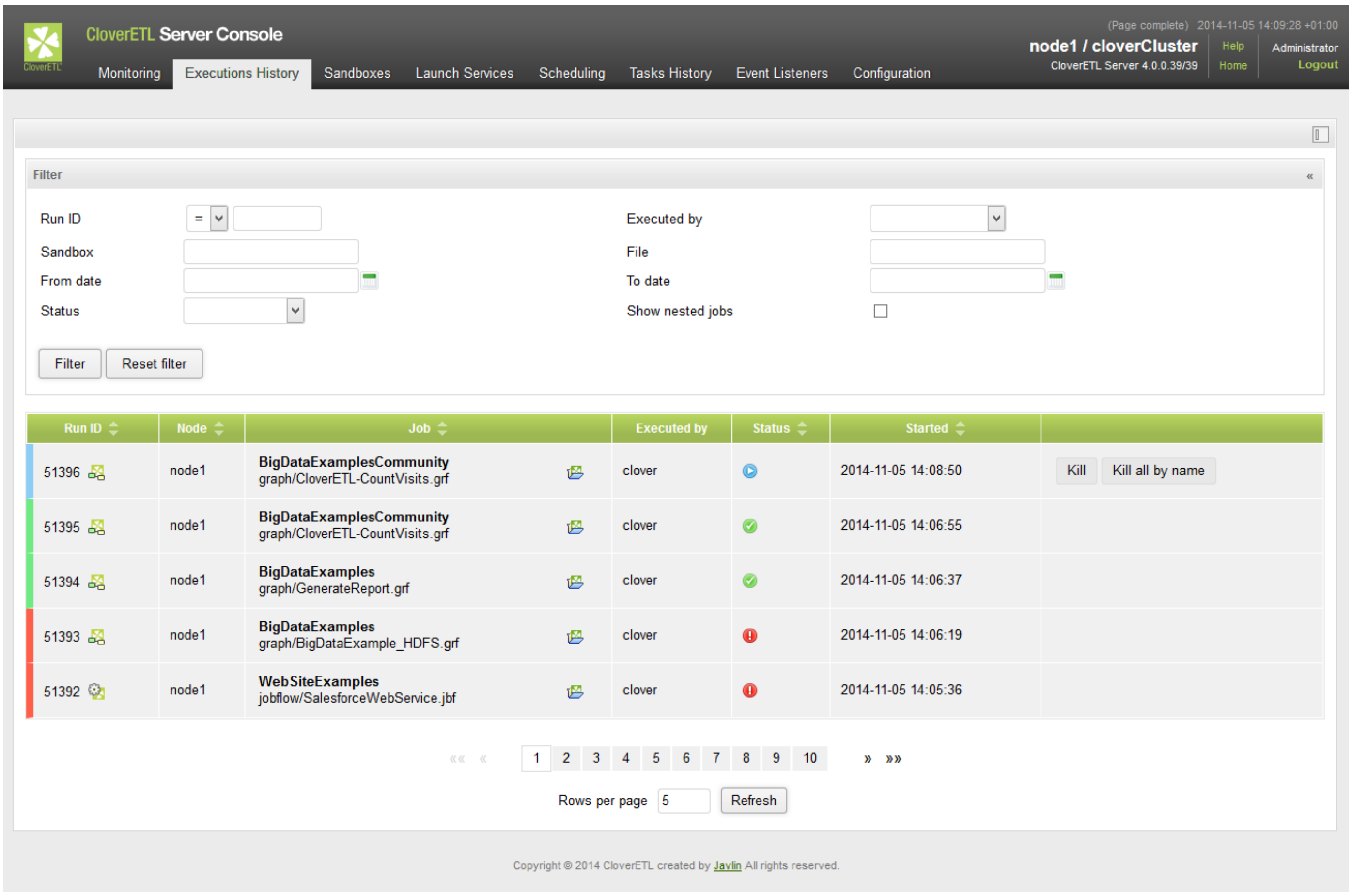
The table shows basic information about the job: Run ID, Job file, current status, and time of execution, as well as some useful links. You will also find additional details after clicking on the job name in the list – details such as associated log files, parameter values, tracking, and more.

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.



Run ID	=	Executed by	
Sandbox		File	
From date		To date	
Status		Show nested jobs	☐

Run ID	Node	Job		Executed by	Status	Started	
51396	node1	**BigDataExamplesCommunity** graph/CloverETL-CountVisits.grf		clover	▶	2014-11-05 14:08:50	Kill · Kill all by name
51395	node1	**BigDataExamplesCommunity** graph/CloverETL-CountVisits.grf		clover	✓	2014-11-05 14:06:55	
51394	node1	**BigDataExamples** graph/GenerateReport.grf		clover	✓	2014-11-05 14:06:37	
51393	node1	**BigDataExamples** graph/BigDataExample_HDFS.grf		clover	!	2014-11-05 14:06:19	
51392	node1	**WebSiteExamples** jobflow/SalesforceWebService.jbf		clover	!	2014-11-05 14:05:36	

Copyright © 2014 CloverETL created by Javlin All rights reserved.

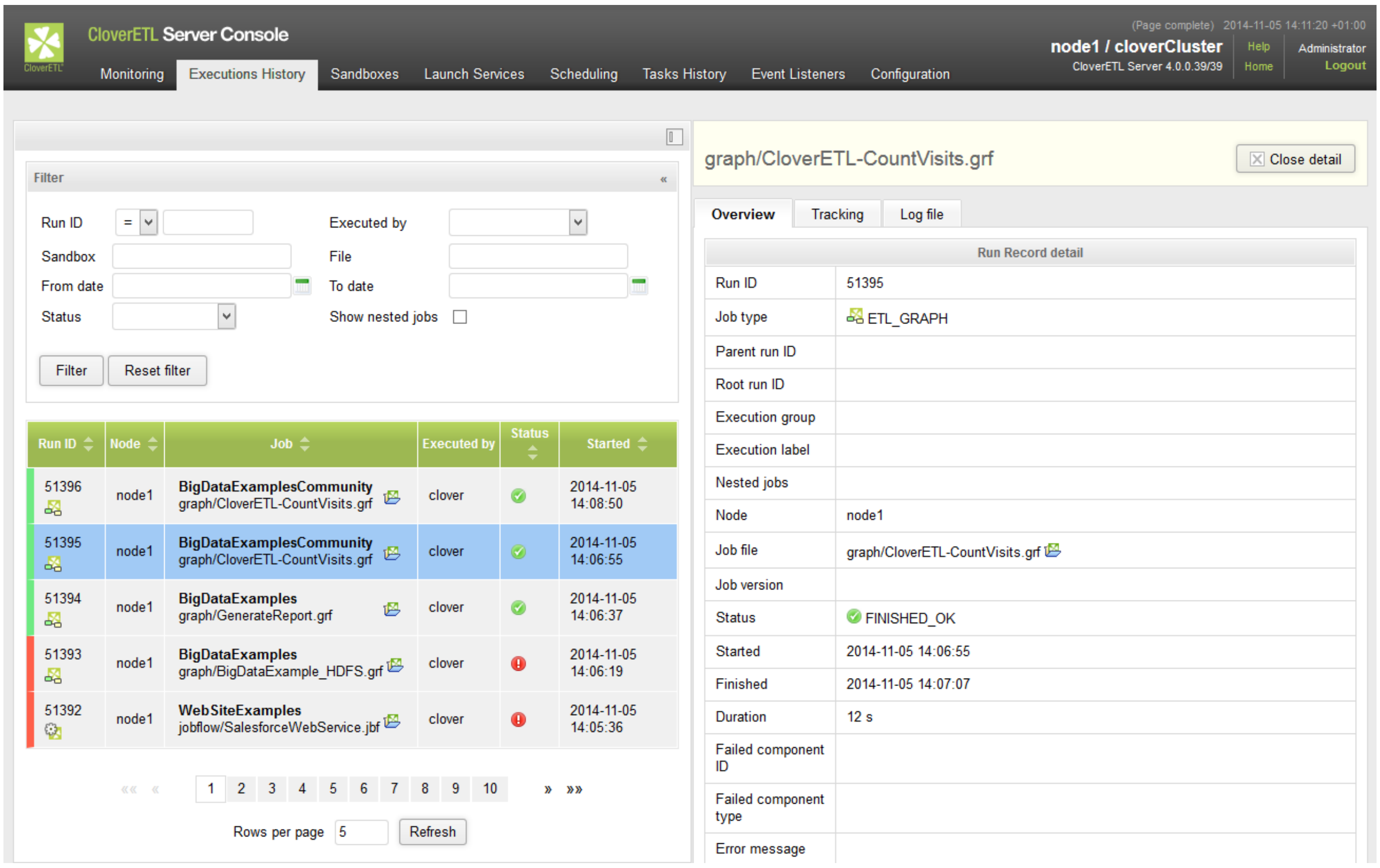
Figure 23.1. Executions History - executions table

When some job execution is selected in the table, the detail info is shown on the right side.

Table 23.1. Persistent run record attributes

Attribute	Description
Run ID	A unique number identifying the run of the job. Server APIs usually return this number as a simple response to the execution request. It's useful as a parameter of subsequent calls for specification of the job execution.
Execution type	Type of a 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 encapsulates 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 a description of Jobflow components 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 a job file. For jobs which are sent together with an execution request, so the job file doesn't exist on the server site, it's set to "default" sandbox.
Job file	Path to a job file, relative to the sandbox root. For jobs which are sent together with an 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 a 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 available) - 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 of 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 an executions hierarchy tree is shown.



Run ID	Node	Job	Executed by	Status	Started
51396	node1	**BigDataExamplesCommunity** graph/CloverETL-CountVisits.grf	clover	✓	2014-11-05 14:08:50
51395	node1	**BigDataExamplesCommunity** graph/CloverETL-CountVisits.grf	clover	✓	2014-11-05 14:06:55
51394	node1	**BigDataExamples** graph/GenerateReport.grf	clover	✓	2014-11-05 14:06:37
51393	node1	**BigDataExamples** graph/BigDataExample_HDFS.grf	clover	!	2014-11-05 14:06:19
51392	node1	**WebSiteExamples** jobflow/SalesforceWebService.jbf	clover	!	2014-11-05 14:05:36

« ‹ 1 2 3 4 5 6 7 8 9 10 › »

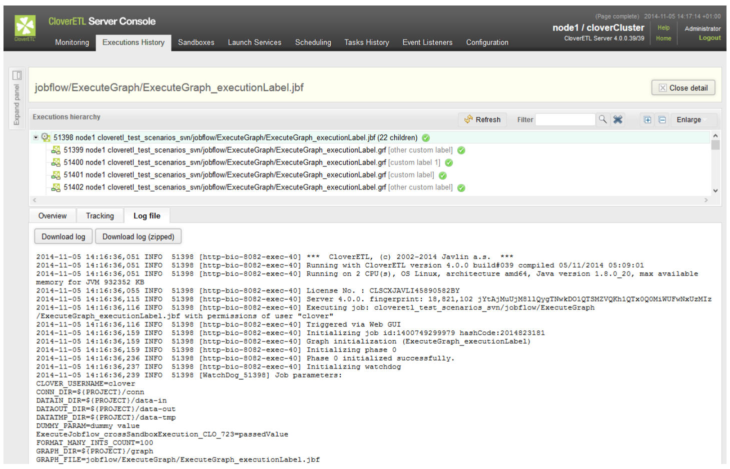
Rows per page 5

graph/CloverETL-CountVisits.grf

Run Record detail	
Run ID	51395
Job type	ETL_GRAPH
Parent run ID	
Root run ID	
Execution group	
Execution label	
Nested jobs	
Node	node1
Job file	graph/CloverETL-CountVisits.grf
Job version	
Status	✓ FINISHED_OK
Started	2014-11-05 14:06:55
Finished	2014-11-05 14:07:07
Duration	12 s
Failed component ID	
Failed component type	
Error message	

Figure 23.2. Executions History - overall perspective

Since the detail panel and expecially job logs may be wide, it may be useful to hide a table on the left, so the detail panel spreads. Click on the minimize icon on the top of the list panel to hide the panel. Then to show list panel again, click to the "Expand panel" icon on the left.



jobflow/ExecuteGraph/ExecuteGraph_executionLabel.jbf

51398 node1 cloveretl_test_scenarios_svn/jobflow/ExecuteGraph/ExecuteGraph_executionLabel.jbf (22 children)
- 51399 node1 cloveretl_test_scenarios_svn/jobflow/ExecuteGraph/ExecuteGraph_executionLabel.grf [other custom label]
- 51400 node1 cloveretl_test_scenarios_svn/jobflow/ExecuteGraph/ExecuteGraph_executionLabel.grf [custom label 1]
- 51401 node1 cloveretl_test_scenarios_svn/jobflow/ExecuteGraph/ExecuteGraph_executionLabel.grf [custom label]
- 51402 node1 cloveretl_test_scenarios_svn/jobflow/ExecuteGraph/ExecuteGraph_executionLabel.grf [other custom label]

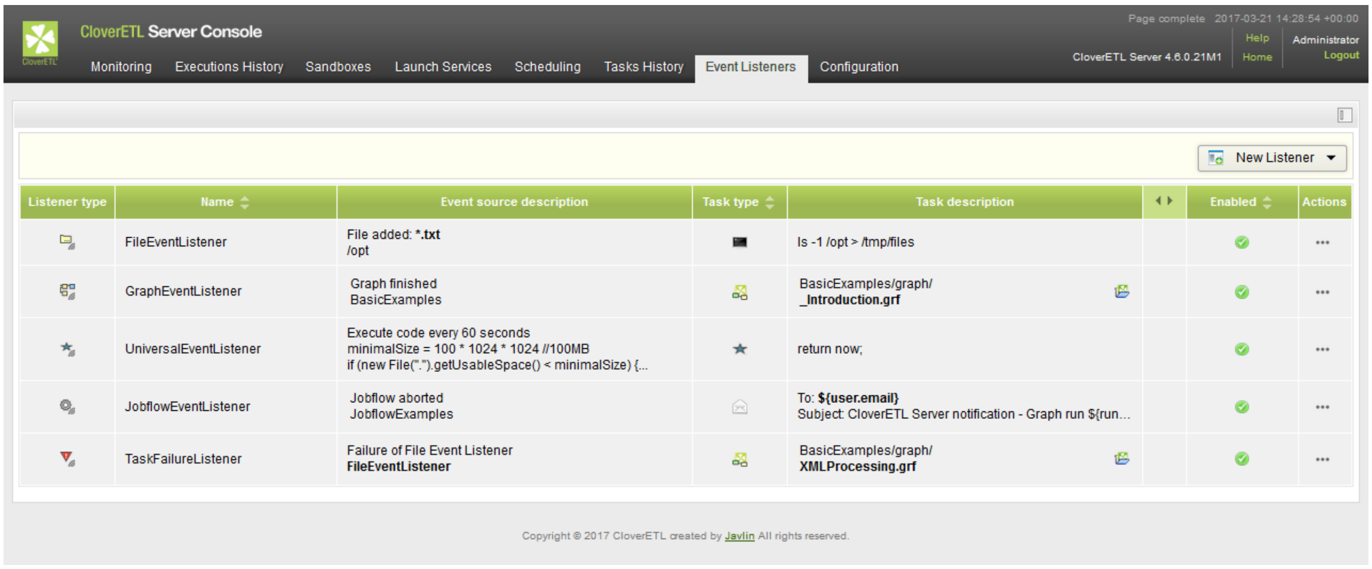
```
2014-11-05 14:16:36,051 INFO  51398 [http-bio-8082-exec-40] ***  CloverETL, (c) 2002-2014 Javlin a.s.  ***
2014-11-05 14:16:36,051 INFO  51398 [http-bio-8082-exec-40] Running with CloverETL version 4.0.0 build#039 compiled 05/11/2014 05:09:01
2014-11-05 14:16:36,051 INFO  51398 [http-bio-8082-exec-40] Running on 2 CPU(s), OS Linux, architecture amd64, Java version 1.8.0_20, max available
memory for JVM 932352 KB
2014-11-05 14:16:36,055 INFO  51398 [http-bio-8082-exec-40] License No. : CLSCXJAVLI45890582BY
2014-11-05 14:16:36,115 INFO  51398 [http-bio-8082-exec-40] Server 4.0.0. fingerprint: 18,821,102 jYtAjMuUjM8l1QygTNwkDO1QTSMZVQKh1QTx0Q0MiWUFwNxUzMIz
2014-11-05 14:16:36,116 INFO  51398 [http-bio-8082-exec-40] Executing job: cloveretl_test_scenarios_svn/jobflow/ExecuteGraph
/ExecuteGraph_executionLabel.jbf with permissions of user "clover"
2014-11-05 14:16:36,116 INFO  51398 [http-bio-8082-exec-40] Triggered via Web GUI
2014-11-05 14:16:36,159 INFO  51398 [http-bio-8082-exec-40] Initializing job id:1400749299979 hashCode:2014823181
2014-11-05 14:16:36,159 INFO  51398 [http-bio-8082-exec-40] Graph initialization (ExecuteGraph_executionLabel)
2014-11-05 14:16:36,159 INFO  51398 [http-bio-8082-exec-40] Initializing phase 0
2014-11-05 14:16:36,236 INFO  51398 [http-bio-8082-exec-40] Phase 0 initialized successfully.
2014-11-05 14:16:36,237 INFO  51398 [http-bio-8082-exec-40] Initializing watchdog
2014-11-05 14:16:36,239 INFO  51398 [WatchDog_51398] Job parameters:
CLOVER_USERNAME=clover
CONN_DIR=${PROJECT}/conn
DATAIN_DIR=${PROJECT}/data-in
DATAOUT_DIR=${PROJECT}/data-out
DATATMP_DIR=${PROJECT}/data-tmp
DUMMY_PARAM=dummy value
ExecuteJobflow_crossSandboxExecution_CLO_723=passedValue
FORMAT_MANY_INTS_COUNT=100
GRAPH_DIR=${PROJECT}/graph
GRAPH_FILE=jobflow/ExecuteGraph/ExecuteGraph_executionLabel.jbf
```

Figure 23.3. 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 a fulltext filter. However when the filter is used, the selected executions aren't hierarchically structured.

Chapter 24. Listeners

Listeners can be seen as hooks. They wait for a specific event and take a used-defined action if the event occurs.



Listener type	Name	Event source description	Task type	Task description		Enabled	Actions
	FileEventListener	File added: ***.txt** /opt		ls -1 /opt > /tmp/files		✓	•••
	GraphEventListener	Graph finished BasicExamples		BasicExamples/graph/ **_Introduction.grf**		✓	•••
	UniversalEventListener	Execute code every 60 seconds minimalSize = 100 * 1024 * 1024 //100MB if (new File(".").getUsableSpace() < minimalSize) {...	★	return now;		✓	•••
	JobflowEventListener	Jobflow aborted JobflowExamples		To: **${user.email}** Subject: CloverETL Server notification - Graph run ${run...		✓	•••
	TaskFailureListener	Failure of File Event Listener **FileEventListener**		BasicExamples/graph/ **XMLProcessing.grf**		✓	•••

Figure 24.1. Listeners

The event is specific to the particular listener

Graph Event Listeners (p. 195)
Jobflow Event Listeners (p. 202)
JMS Messages Listeners (p. 205)
Universal Event Listeners (p. 210)
File Event Listeners (remote and local) (p. 212)
Task Failure Listeners (p. 218)

The available actions taken by the listeners are common for all listeners. The action, that can be taken are:

- Send an e-mail - see Send an Email (p. 163)
- Execute a shell command - see Execute Shell Command (p. 167)
- Start a graph - see Start a Graph (p. 170)
- Start a jobflow - see Start a Jobflow (p. 172)
- Start a profiler job
- Abort a job - see Abort job (p. 174)
- Archivator - see Archive Records (p. 175)
- Send a JMS Message - see Send a JMS Message (p. 177)
- Execute a Groovy code - see Execute Groovy Code (p. 180)

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. 202)) – 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 a graph, which is source of the event. If there is an event listener specified, task may use these properties. E.g. next graphs in the chain may use "EVENT_FILE_NAME" placeholder which was activated by the first graph in the chain. Graph properties, which are set specifically for each graph run (i.e. RUN_ID), are overridden by the last graph.

Types of graph events

- Graph started (p. 195)
- Graph phase finished (p. 195)
- Graph finished (p. 195)
- Graph error (p. 195)
- Graph aborted (p. 196)
- Graph timeout (p. 196)
- Graph unknown status (p. 196)

Graph started

Graph started event is created, when ETL graph execution successfully started.

Graph phase finished

Graph phase finished event is created, everytime when graph phase is finished and all its nodes are finished with status `FINISHED_OK`.

Graph finished

Graph finished event is created, when all phases and nodes of graph are finished with `FINISHED_OK` status.

Graph error

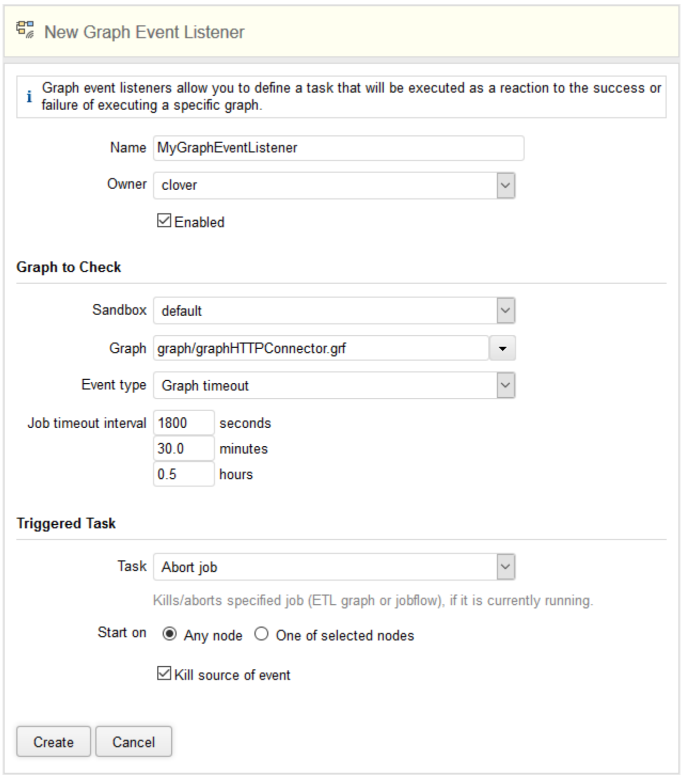
Graph error event is created, when graph cannot be executed for any reason, or when any node of graph fails.

Graph aborted

Graph aborted event is created, when graph is explicitly aborted.

Graph timeout

Graph timeout event is created, when graph runs longer than for a specified interval. Thus you should specify a "Job timeout interval" attribute for each listener of a graph timeout event. You can specify this interval in seconds or in minutes or in hours.


New Graph Event Listener

Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name: MyGraphEventListener
Owner: clover
[✓] Enabled

Graph to Check

Sandbox: default
Graph: graph/graphHTTPConnector.grf
Event type: Graph timeout
Job timeout interval: 1800 seconds
30.0 minutes
0.5 hours

Triggered Task

Task: Abort job
Kills/aborts specified job (ETL graph or jobflow), if it is currently running.
Start on: [✓ Any node] One of selected nodes
[✓] Kill source of event

Figure 24.2. Web GUI - graph timeout event

Graph unknown status

Graph unknown status event 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 the graph run. Server automatically changes state of a graph to *Not Available* and sends a *graph unknown status* event.

Please note, that this works just for executions, which have a persistent record in executions history. It is possible to execute a transformation without persistent record in executions history, typically for better performance of fast running transformations (i.e. using Launch Services).

Listener

User may create a listener for a specific event type and graph (or all graphs in sandbox). The listener is actually connection between graph event and the task, where graph event specifies *when* and task specifies *what* to do.

Event handling consist of the following course of actions:

- event is created

- listeners for this event are notified
- each listener performs related task

Tasks

Task types are described in Chapter 20, Tasks (p. 162), see this chapter for details about these task types.

In the Cluster environment, all tasks have an additional attribute "Node IDs to process the task". If there is no node ID specified, the task may be processed on any cluster node. In most cases it will be processed on the same node where the event was triggered. If there are some nodeIDs specified, the task will be processed on the first node in the list which is connected in cluster and ready.

Send an Email (p. 163)
Execute Shell Command (p. 167)
Start a Graph (p. 170)
Start a Jobflow (p. 172)
Start a Profiler Job (p. 173)
Abort job (p. 174)
Archive Records (p. 175)
Send a JMS Message (p. 177)
Execute Groovy Code (p. 180)

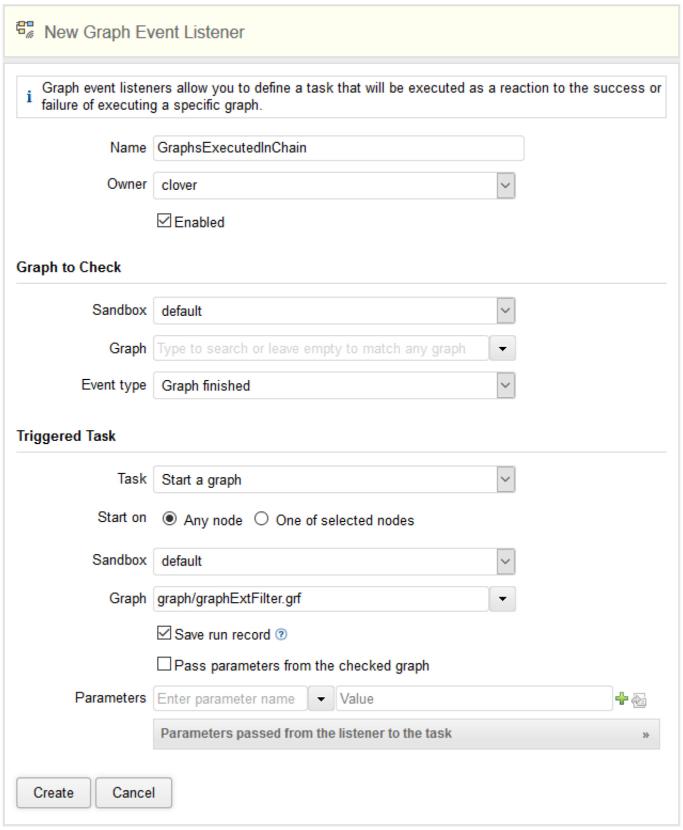
Use Cases

Possible use cases are the following:

- Execute graphs in chain (p. 197)
- Email notification about graph failure (p. 199)
- Email notification about graph success (p. 200)
- Backup of data processed by graph (p. 201)

Execute graphs in chain

For example, 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 a graph event listener. We create a listener for `graph finished OK` event of graph A and choose an `execute graph` task type with graph B specified for execution. If we create another listener for graph B with `execute graph` task with graph C specified, it will work as a chain of graphs.


New Graph Event Listener

Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name: GraphsExecutedInChain
Owner: clover
[✓] Enabled

Graph to Check

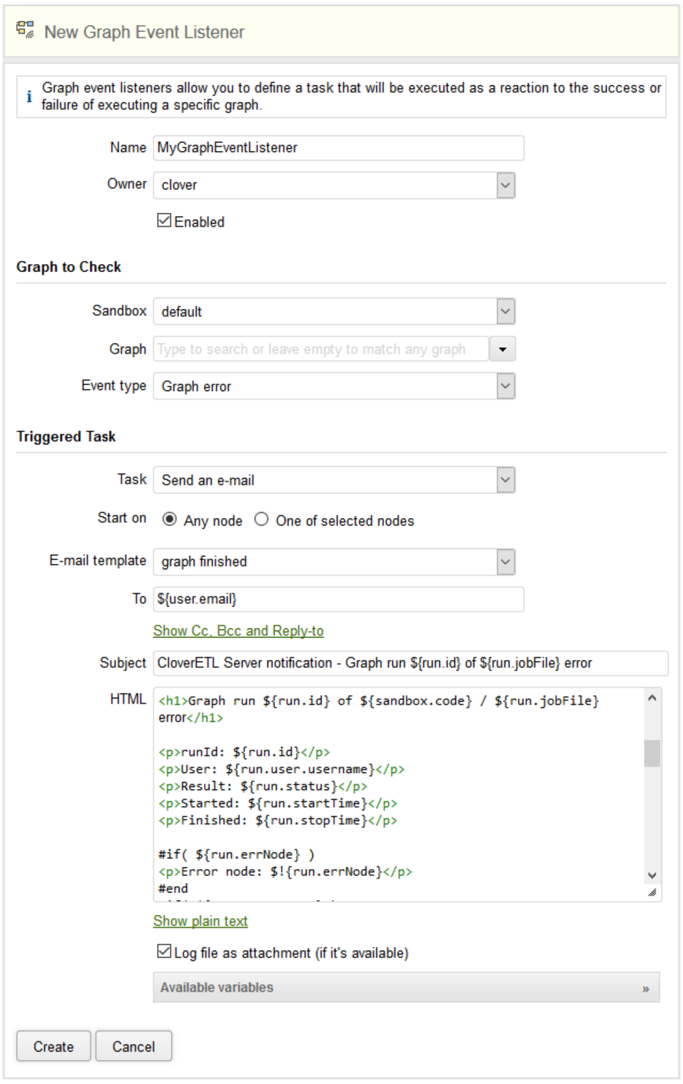
Sandbox: default
Graph: Type to search or leave empty to match any graph
Event type: Graph finished

Triggered Task

Task: Start a graph
Start on: [✓ Any node] One of selected nodes
Sandbox: default
Graph: graph/graphExtFilter.grf
[✓] Save run record
[ ] Pass parameters from the checked graph
Parameters: Enter parameter name | Value
Parameters passed from the listener to the task

Figure 24.3. Event source graph isn't specified, thus listener works for all graphs in specified sandbox

Email notification about graph failure



New Graph Event Listener

Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name	MyGraphEventListener
Owner	clover
	☑ Enabled

Graph to Check

Sandbox	default
Graph	Type to search or leave empty to match any graph
Event type	Graph error

Triggered Task

Task	Send an e-mail
Start on	[✓ Any node] One of selected nodes
E-mail template	graph finished
To	${user.email}
	Show Cc, Bcc and Reply-to
Subject	CloverETL Server notification - Graph run ${run.id} of ${run.jobFile} error

HTML

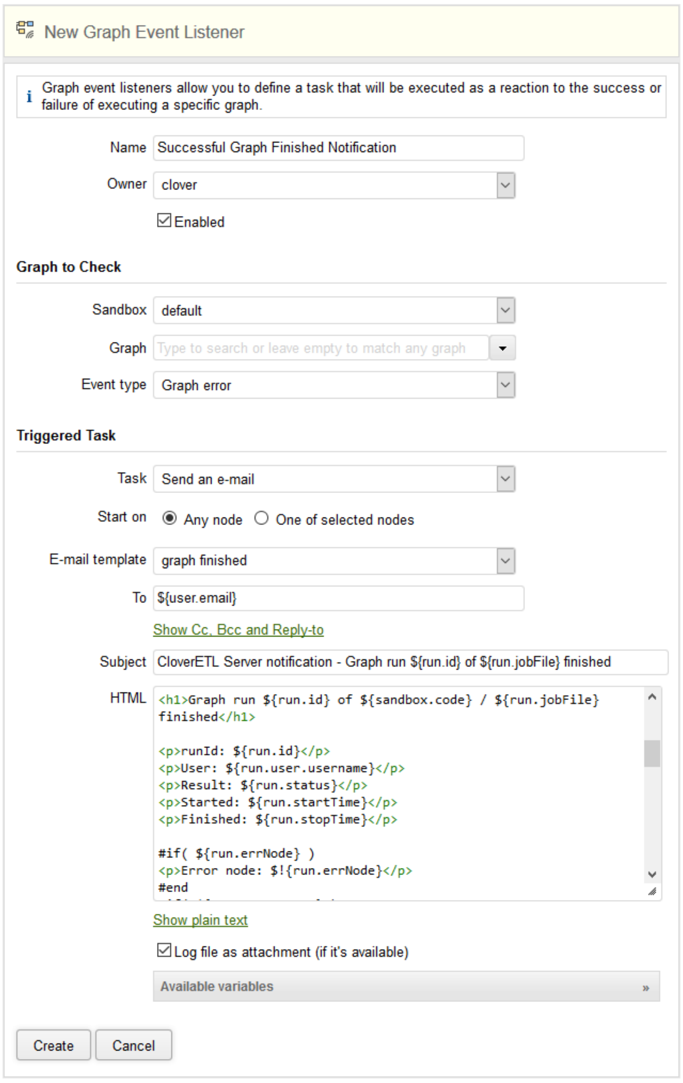
```
<h1>Graph run ${run.id} of ${sandbox.code} / ${run.jobFile}
error</h1>

<p>runId: ${run.id}</p>
<p>User: ${run.user.username}</p>
<p>Result: ${run.status}</p>
<p>Started: ${run.startTime}</p>
<p>Finished: ${run.stopTime}</p>

#if( ${run.errNode} )
<p>Error node: $!{run.errNode}</p>
#end
```

Show plain text

☑ Log file as attachment (if it's available)

Figure 24.4. Web GUI - e-mail notification about graph failure

Email notification about graph success



Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name	Successful Graph Finished Notification
Owner	clover
	☑ Enabled

Graph to Check

Sandbox	default
Graph	Type to search or leave empty to match any graph
Event type	Graph error

Triggered Task

Task	Send an e-mail
Start on	● Any node ○ One of selected nodes
E-mail template	graph finished
To	${user.email}
	Show Cc, Bcc and Reply-to
Subject	CloverETL Server notification - Graph run ${run.id} of ${run.jobFile} finished

HTML

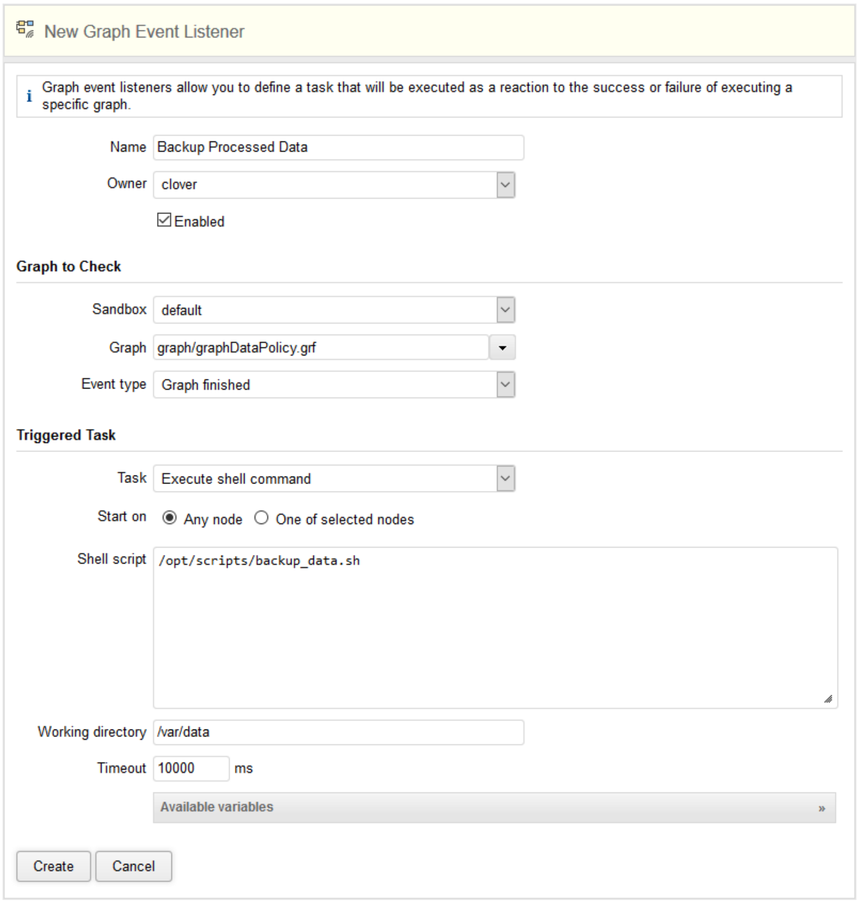
```
<h1>Graph run ${run.id} of ${sandbox.code} / ${run.jobFile}
finished</h1>

<p>runId: ${run.id}</p>
<p>User: ${run.user.username}</p>
<p>Result: ${run.status}</p>
<p>Started: ${run.startTime}</p>
<p>Finished: ${run.stopTime}</p>

#if( ${run.errNode} )
<p>Error node: $!{run.errNode}</p>
#end
```

Show plain text

☑ Log file as attachment (if it's available)

Figure 24.5. Web GUI - email notification about graph success

Backup of data processed by graph


New Graph Event Listener

Graph event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific graph.

Name	Backup Processed Data
Owner	clover
	☑ Enabled

Graph to Check

Sandbox	default
Graph	graph/graphDataPolicy.grf
Event type	Graph finished

Triggered Task

Task	Execute shell command
Start on	[✓ Any node] One of selected nodes
Shell script	/opt/scripts/backup_data.sh
Working directory	/var/data
Timeout	10000 ms

Figure 24.6. 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 (Graph Event Listeners (p. 195)) 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 the first job in the chain. Job properties, which are set specifically for each run (e.g. RUN_ID), are overridden by the last job.

Types of jobflow events

- Jobflow started (p. 202)
- Jobflow phase finished (p. 202)
- Jobflow finished (p. 202)
- Jobflow error (p. 202)
- Jobflow aborted (p. 202)
- Jobflow timeout (p. 202)
- Jobflow unknown status (p. 203)

Jobflow started

A **Jobflow started** event is created, when jobflow execution successfully started.

Jobflow phase finished

A **Jobflow phase finished** event is created, everytime when jobflow phase is finished and all its nodes are finished with status `FINISHED_OK`.

Jobflow finished

A **Jobflow finished** event is created, when all phases and nodes of jobflow are finished with status `FINISHED_OK`.

Jobflow error

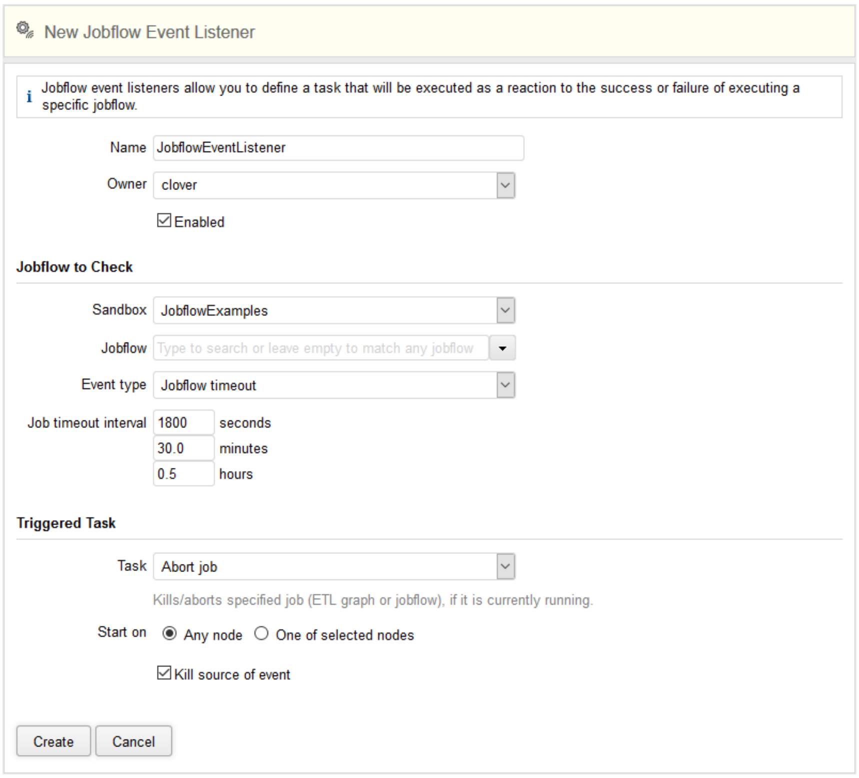
A **Jobflow error** event is created, when jobflow cannot be executed from any reason, or when any node of the jobflow fails.

Jobflow aborted

A **Jobflow aborted** event is created, when jobflow is explicitly aborted.

Jobflow timeout

A **Jobflow timeout** event 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.


New Jobflow Event Listener

Jobflow event listeners allow you to define a task that will be executed as a reaction to the success or failure of executing a specific jobflow.

Name: JobflowEventListener
Owner: clover
☑ Enabled

Jobflow to Check

Sandbox: JobflowExamples
Jobflow: Type to search or leave empty to match any jobflow
Event type: Jobflow timeout
Job timeout interval: 1800 seconds, 30.0 minutes, 0.5 hours

Triggered Task

Task: Abort job
Kills/aborts specified job (ETL graph or jobflow), if it is currently running.
Start on: [✓ Any node] One of selected nodes
☑ Kill source of event

Figure 24.7. Web GUI - jobflow timeout event

Jobflow unknown status

A **Jobflow unknown status** event 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 a listener for the specified event type and jobflow (or all jobflows in sandbox). The listener is actually connection between jobflow event and task, where jobflow event specifies *when* and task specifies *what* to do.

Event handling consist of the following course of actions:

- event is created
- listeners for this event are notified
- each listener performs the related task

Tasks

Task specifies operation which should be performed as the reaction to the triggered event.

Task types are described in Chapter 20, Tasks (p. 162).

Note: You can use a task of any type for a jobflow event listener. Description of task types is divided into two sections just to show the most obvious use cases.

Send an Email (p. 163)
Execute Shell Command (p. 167)
Start a Graph (p. 170)
Start a Jobflow (p. 172)
Start a Profiler Job (p. 173)
Abort job (p. 174)
Archive Records (p. 175)
Send a JMS Message (p. 177)
Execute Groovy Code (p. 180)

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 24.1. Attributes of JMS message task

Attribute	Description
Initialize by	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" consume messages concurrently. • Failover with node reservation: More node IDs specified (separated by a 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 a standalone environment, the "Node IDs" attribute is ignored.
JNDI Access	
Initial context	Default or custom
Initial context factory class	A full class name of javax.naming.InitialContext implementation. Each JMS provider has its own implementation. E.g. Apache MQ has "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 a JMS API implementation for each specific JMS broker provider.
Broker URL	URL of a JMS message broker

Attribute	Description
Listen To	
Connection factory	JNDI name of a connection factory. It depends on a JMS provider.
Username	Username for connection to a JMS message broker
Password	Password for connection to JMS message broker
Queue/Topic	JNDI name of a message queue/topic on the server
Durable subscriber	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 the attribute is true, the 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 on JMS provider. E.g. for ActiveMQ, it is set as a 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.
Message Processing	
Number of consumers	E.g. 1
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 24.2. Variables accessible in groovy code

type	key	description
javax.jms.Message	msg	instance of a JMS message
java.util.Properties	properties	See below for details. It contains values (String or converted to String) read from a message and it is passed to the task which may use them somehow. E.g. "execute graph" task 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. E.g. "execute graph" task passes it to the executed graph as "dictionary entries".
javax.servlet.ServletContext	servletContext	instance of ServletContext
com.cloveretl.server.api.ServerFacade	serverFacade	instance of serverFacade usable for calling CloverETL Server core features.
java.lang.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 into two data structures: Properties and Data.

Table 24.3. Properties Elements

key	description
JMS_PROP_[property key]	For each message property is created one entry, where "key" is made of a "JMS_PROP_" prefix 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 "JMS_MAP_" prefix 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 a provider-specific message ID or an 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 24.4. "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 24.5. 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 the 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" concurrently perform Groovy code. So the code is executed on each node in the 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 Criteria

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 is considered a coding error and the event is NOT triggered. Thus, exceptions should be properly handled in the Groovy code.

Table 24.6. Variables accessible in Groovy code

type	key	description
java.util.Properties	properties	An 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	An 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". 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 in which CloverETL Server is running
com.cloveretl.server.api.ServerFacade	serverFacade	instance of serverFacade usable for calling CloverETL Server core features.
java.lang.String	sessionToken	sessionToken, needed for calling methods on the serverFacade

File Event Listeners (remote and local)

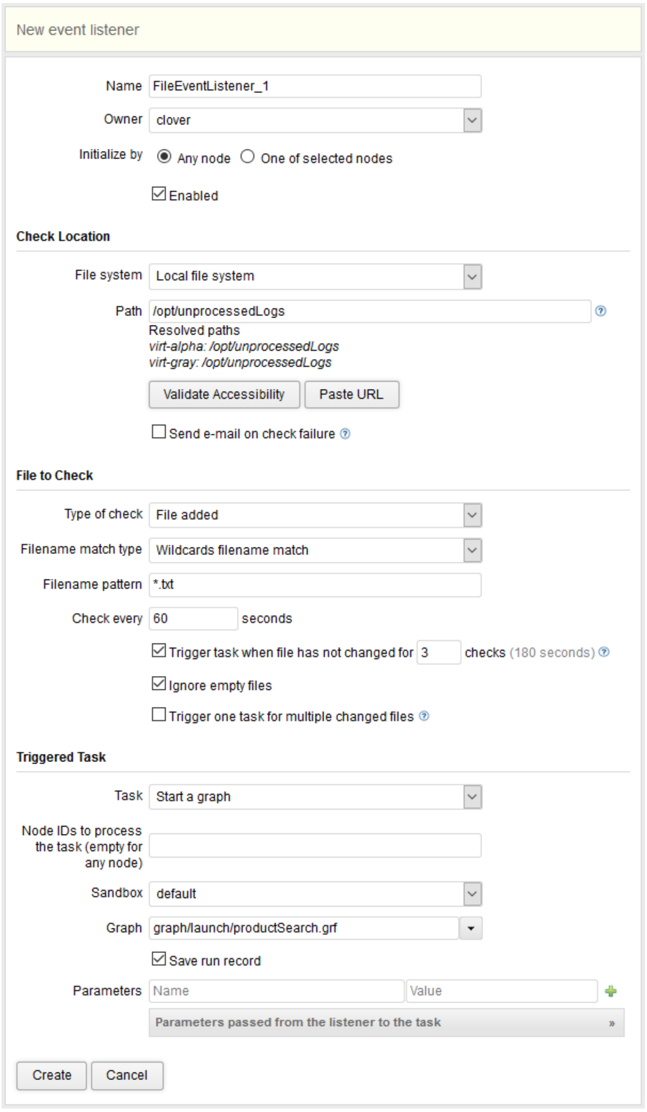
Local file-system changes: Since 1.3

Remote file-system changes: Since 4.2

File Event Listeners allow you to monitor changes on a specific local file system path or remote URL – for example, new files appearing in a folder – and react to such an event with a predefined task.

You can either specify an exact file name or use a wildcard or regexp, then set a checking interval in seconds, and finally, define a task to process the event.

There is a global minimum check interval that you can change if necessary in the configuration ("clover.event.fileCheckMinInterval" property). See Chapter 9, List of Properties (p. 88).


New event listener

Name	FileEventListener_1
Owner	clover
Initialize by	[✓ Any node] One of selected nodes
	☑ Enabled

Check Location

File system	Local file system
Path	/opt/unprocessedLogs
	Resolved paths *virt-alpha: /opt/unprocessedLogs* *virt-gray: /opt/unprocessedLogs*
	☐ Send e-mail on check failure

File to Check

Type of check	File added
Filename match type	Wildcards filename match
Filename pattern	*.txt
Check every	60 seconds
	☑ Trigger task when file has not changed for 3 checks (180 seconds)
	☑ Ignore empty files
	☐ Trigger one task for multiple changed files

Triggered Task

Task	Start a graph
Node IDs to process the task (empty for any node)	
Sandbox	default
Graph	graph/launch/productSearch.grf
	☑ Save run record
Parameters	Name / Value
	Parameters passed from the listener to the task

Figure 24.8. Web GUI - creating a File Event listener

Cluster environment

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 observes the local/remote filesystem, however node status must be "ready". When the node isn't ready, file system isn't checked at all.

  To create a file event listener with no failover, select **One of selected nodes** in **Initialize by** and select one node from the table below.

- **Failover with node concurrency**: No node ID specified (empty input) - All cluster nodes with status "ready" concurrently check local/remote filesystem according to file event listener attributes settings. In this mode, when the listener is configured to observe local filesystem, each cluster node observes its own local file system. So it's useful only when the observed path is properly shared among the cluster nodes. It may behave unpredictably otherwise. On the other hand, when the listener is configured to observe remote filesystem, listeners running on different cluster nodes may connect to the same remote resource. The nodes use locking mechanism when accessing the local or remote filesystem, so the listeners running concurrently on different nodes can't get to the conflict.

  To create file event listener with node cuncurrency, select **Any node** in **Initialize by**.

- **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 isn't "ready"), any other "ready" node from the list continues with checking. Please note, that when file event listener is re-initialized on another cluster node, it compares last directory content detected by the failed node with the current directory content.

  To create a file event listener with node reservation, select **One of selected nodes** in **Initialize by** and select more nodes.

In standalone environment, the "Node IDs" attribute is ignored.

Supported filesystems and protocols

Local filesystem

The user may specify path to the directory which the listener shall observe. The listener doesn't read the directory content recursivelly. The directory must exist.

In cluster environment the directory must exist on each cluster node where the listener may run. If the listener can run concurrently on more nodes, the directory must be shared among all these nodes and the directory must exist on all these nodes.

It's recommended to use placeholders to unify configuration on all nodes. These are recommended placeholders: CloverETL Server config property `${sandboxes.home}` and JVM system property `${java.io.tmpdir}`. It's possible to use any JVM system property or Environment variable.

Remote filesystem

The user may specify URL to the directory which the listener shall observe. Currently these protocols are supported: FTP, S3, SFTP and SMB. Different protocols may use different authentication methods: none, username+password and keystore. The listener doesn't read the directory content recursivelly. The directory must exist.

Currently the subset of the protocols allowed by file-operations is supported:

- **FTP - File Transfer Protocol** (no authentication or username+password authentication) URL example:

```
            ftp://host:23/observed/path/
```

- **SFTP (SSH/FTP) - SSH File Transfer Protocol** (username+private key authentication) URL example:

```
sftp://host:23/observed/path/
```

It's recommended to use placeholders to unify path configuration on all nodes. These are recommended placeholders: CloverETL Server config property `${sandboxes.home}`, JVM system property `${user.home}`. It's possible to use any JVM system property or Environment variable.

- **S3 - Amazon S3 Storage** (AWSAccessKeyId+Signature authentication) URL example:

```
s3://s3.amazonaws.com/bucketname/path/
```

Please specify the AWSAccessKeyId as username and Signature as password.

- **SMB - Microsoft SMB Protocol** (username+password authentication) URL example:

```
smb://host/path/
```

Observed file

Local observed file is specified by directory path and file name pattern.

Remote observed file is specified by URL, credentials 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. 214)
- Wildcards (p. 214)
- Regular expression (p. 215)

Exact match

You specify the 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 with placeholders. It is possible to use relative path, but working directory depends on an 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.

Please note that since CloverETL 4.2, the grouping mode may be enabled for the file listener, so all file changes detected by single check produce just one "grouped" file event. Otherwise each single file produces its own event.

There are four types of file events:

- File added (p. 215)
- File removed (p. 215)
- File size changed (p. 215)
- File timestamp changed (p. 215)

File added

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 a new file is detected, regardless it is complete or not. Thus task which may need a complete file is executed when file is still incomplete. Recommended approach is to save the file to a different location and when it is complete, rename it or move it to an 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 the previous check and it exists now, during the current check.

File removed

Event of this type occurs, when observed file is deleted or moved to another location between two checks.

File size changed

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. The file must exist during both checks.

File timestamp changed

Event of this type occurs, when change time of the observed file has changed between two checks. Event of this type is never produced when the file is created or removed. The file must exist during both checks.

Check Interval, Task and Use Cases

- User may specify the minimum time interval between two checks. The interval is specified in seconds. **Check every**

- Each listener defines a task, which will be processed as a reaction for the file event. All task types and theirs attributes are described in Scheduling and GraphEventListeners sections.
    - Graph Execution, when a file with input data is accessible
    - Graph Execution, when a file with input data is updated
    - Graph Execution, when a file with generated data is removed and must be recreated

How to use source of event during task processing

A file(s), which caused event (considered as source of the event) may be used during task processing. CloverETL graph/jobflow components with "File URL" attribute (e.g. reader or writer components) may directly use event source by parameter placeholder: `${EVENT_FILE_URLS}`. Please see Executed by Task "graph execution" by File Event Listener (p. 160) to see another parameters.

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.

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).

Delayed triggering for incomplete files

It is possible to delay task execution for incomplete files. This is useful in cases, when the condition to execute the listener's task has been met, but there is still some work that needs to be done on the file, e.g. the whole file needs to be uploaded.

Ignore empty files

If the process creating the file creates an empty file and does something else for several minutes or even hours, and finally writes the content, tick **Ignore empty files** checkbox. The task will be triggered only if a non-empty file appears.

Trigger task when file has not changed for n checks

If the file size slowly rises till the file is complete, tick the checkbox **Trigger task when file has not changed**. Then specify the number of additional file size checks that are to be performed on the file. The listener's task will not be triggered until the checks are performed and the file's size stays the same between these checks.

Combination

If you use **Ignore empty files** and **Trigger task when file has not changed for n checks** together, the first one filters out empty files and the later one the files that are being changed. The task will be triggered only on files that are not empty and that have not changed for the specified number of checks.

Howtos

Create a file event listener listening to changes on local file system (p. 216)
Observe file from one cluster node (p. 217)
Quickly setup failure notification (p. 217)
Quickly enable or disable file event listener (p. 217)

Create a file event listener listening to changes on local file system

This howto shows way to create a new listener checking appearance of a file (`new_invoices.txt`) on local file system (`/mnt/sdb2/`). The appearance will trigger a graph (`graph/checkInvoices.grf`) from **InvoicesProcessing** sandbox.

In **Event Listeners** →**File Event Listeners**, click **New Listener**.

Enter the **Name** of the listener, e.g. **Invoices**.

Enter the **Path** to the directory where files will appear: `/mnt/sdb2`. You can check that Clover can access this directory (the directory exists and permissions are set up properly) with **Validate Accesibility** button.

If the observed directory becomes inaccessible, CloverETL Server can send you an email. To do so, tick **Send email on check failure** and enter recipient(s).

The event should be triggered on file appearance - set **Type of check** to **File added**.

Enter the file name `new_invoices.txt` to **Filename pattern**.

If the file is created empty but the content is written after some time, tick **Ignore empty files**. Doing so, the task will be executed after the file contains some data.

If it takes a long time to copy the whole file to the observed position, the clover server can perform several check to ensure that the file to process is not to be changed. Tick **Trigger task when file has not changed for** and enter the number of checks. If you tick **Ignore empty files**, this checks will be performed after the file is not empty.

Choose **Sandbox** with the graph (**InvoicesProcessing**) and the **graph** (**graph/checkInvoices.grf**).

To save the changes click **Create**.

Observe file from one cluster node

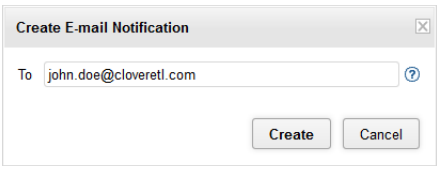
Create the listener in the same way as on Server.

Switch **Initialize by** to **One of selected nodes**.

Add the particular node(s) from **Available nodes** to **Selected nodes**.

Quickly setup failure notification

To create notification when the file event listener fails, click **Create notification** button. Pressing the button opens up a popup dialog where e-mail addresses can be entered separated by commas.


Create E-mail Notification

To: john.doe@cloveretl.com

The entered email addresses are remembered and pre-filled the next time the button is pressed. If the popup is closed with invalid e-mail addresses entered, then the field is cleared.

When creating the notification a Task Failure Listener is created with an e-mail task listening to the selected File Event Listener. The first entered e-mail address will be used as the Reply-to(Sender) address. The subject and body of the e-mail is as predefined by the Task Failure template. The trigger limit is set to 5.

Editing failure notification

If there is a Task Failure Listener listening to given File Event Listener then instead of Create Notification button a Notification Detail button is displayed. This button redirects to the Task Failure Listener page and shows the details of the Task Failure Listener listening to the File Event Listener. If more than one Task Failure Listeners are listening to the File Event Listener, then the details of the first one is shown.

Quickly enable or disable file event listener

In **Event Listeners** →**File Event Listeners**, there is a table with event listeners. In this table, click the icon in **Enabled** column.

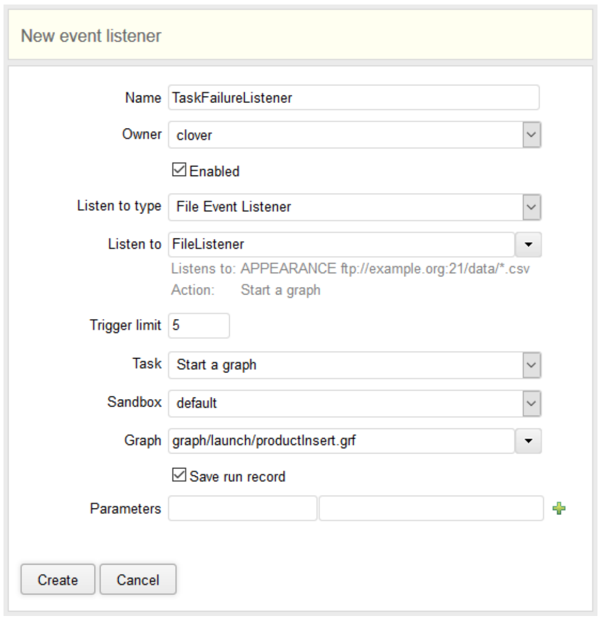
Task Failure Listeners

Since 4.4

Task Failure Listeners allow you to detect and react to failures in your server when a task you set up in a listener fails to execute, e.g. a File Listener is set up to detect changes on an FTP server, but it fails to connect to the FTP server.

Task Failure Listeners do not detect failures of the task itself, e.g. a File Listener is set up to detect changes on an FTP server and send an e-mail if the change is detected. If the File Listener fails to send the e-mail for some reason, the Task Failure Listener won't detect it.

The same tasks to be executed are available as with all the other listeners, with the difference being that when creating a new Task Failure Listener the pre-selected task is Sending an e-mail if the e-mail service is configured in Configuration.



New event listener

Name	TaskFailureListener
Owner	clover
	☑ Enabled
Listen to type	File Event Listener
Listen to	FileListener
	Listens to: APPEARANCE ftp://example.org:21/data/*.csv
	Action: Start a graph
Trigger limit	5
Task	Start a graph
Sandbox	default
Graph	graph/launch/productInsert.grf
	☑ Save run record
Parameters	

Figure 24.9. Web GUI - creating a Task Failure listener

Task Choice

There are three options to choose from: Listening to any task failure, listening to failures from a group (such as File Event Listeners), or listening to a failure of a chosen listener.

Selecting an option from the 'Listen to type' menu restricts the 'Listen to' combobox to event sources of the chosen category. If there are no event sources in the category, the Failure Listener still can be created, it will react to failures of tasks that will be created in that category.

When selecting an event source you can type into the text field to filter the dropdown menu. After selecting an event source some information is presented about the listener.

When sending an e-mail you can use the 'task failed' template by selecting it from the 'E-mail template' dropdown menu.

Task Failed E-mail Template

The default e-mail template may be modified using placeholders described in Placeholders (p. 164) and parameters in Table 24.7, "Parameters usable in task failure e-mail template" (p. 219). Furthermore, some additional parameters can be used if the failed listener is a File Event Listener, see Table 24.8, “File Event Listener specific parameters usable in task failure e-mail template” (p. 219).

Table 24.7. Parameters usable in task failure e-mail template

Parameter	Description
TASK_LISTENER_ID	ID of the failed listener.
TASK_LISTENER_NAME	Name of the failed listener.
TASK_LISTENER_TYPE	Type of the failed listener.
TASK_LISTENER_TYPE_TEXT	Full name of the failed listener's type.
TASK_LISTENER_OWNER_USERNAME	Owner of the failed listener.

Table 24.8. File Event Listener specific parameters usable in task failure e-mail template

Parameter	Description
FILE_EVENT_LISTENER_FILE_PATH	Path to the observed directory.
FILE_EVENT_LISTENER_FILE_REMOTE_URL	Full remote URL of the observed directory.
FILE_EVENT_LISTENER_FILE_NAME_PATTERN	Filename pattern of the files the listener observes.
FILE_EVENT_LISTENER_FILE_CHECK_TYPE	The type of check the listener performs.
FILE_EVENT_LISTENER_FILE_MATCH_TYPE	The listener's filename match type.
FILE_EVENT_LISTENER_NODE	Node IDs of nodes the listener can be intitalized by.

Chapter 25. 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 (p. 220)
- Operation graph_run (p. 221)
- Operation graph_status (p. 221)
- Operation graph_kill (p. 222)
- Operation server_jobs (p. 223)
- Operation sandbox_list (p. 223)
- Operation sandbox_content (p. 223)
- Operation executions_history (p. 223)
- Operation suspend (p. 225)
- Operation resume (p. 225)
- Operation sandbox_create (p. 226)
- Operation sandbox_add_location (p. 226)
- Operation sandbox_remove_location (p. 226)
- Operation download_sandbox_zip (p. 227)
- Operation upload_sandbox_zip (p. 227)
- Operation cluster_status (p. 228)
- Operation export_server_config (p. 228)
- Operation import_server_config (p. 229)

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. The operation is called graph_run for backward compatibility, however it may execute ETL graph, jobflow or profiler job.

parameters

Table 25.1. Parameters of graph_run

parameter name	mandatory	default	description
graphID	yes	-	A 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 a placeholder, but without the "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. 137) for more info.
nodeID	no	-	In cluster mode it's ID of a 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. The operation is called graph_status for backward compatibility, however it may return status of an ETL graph or jobflow.

parameters

Table 25.2. 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 the graph is in the required status.
waitTimeout	no	0	If waitForStatus is specified, it will wait only the specified amount of milliseconds. Default 0 means forever, but it depends on an application server configuration. When the specified timeout expires and graph run still isn't in required status, the server returns code 408 (Request Timeout). 408 code may be also returned by an application server if its HTTP request timeout expires before.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should possible error message be.

returns

Status of a specified graph. It may be number code, text code or a complex description in dependence on optional parameter returnType. Description is returned as plain text with a pipe as a separator, or as XML. A 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 it may return result immediately or wait for a 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. The operation is called graph_kill for backward compatibility, however it may abort/kill ETL graph, jobflow or profiler job.

parameters

Table 25.3. 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 the specified graph after an attempt to kill it. It may be number code, text code or a complex description in dependence on optional parameter.

example

```
http://localhost:8080/clover/request_processor/graph_kill?runID=123456&returnType=DESCRIPTION
```

Operation server_jobs

parameters

no

returns

List of runIDs of 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 25.4. 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

A list of all elements in the specified sandbox. Each element may be specified as a file path relative to the sandbox root.

example

```
http://localhost:8080/clover/request_processor/sandbox_content?sandbox=mva
```

Operation executions_history

parameters

Table 25.5. Parameters of executions_history

parameter name	mandatory	default	description
sandbox	yes	-	text ID of sandbox
from	no		Lower datetime limit of start of execution. The 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. The 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. The 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. The 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. The operation will return only records with specified STATUS. Meaningful values are RUNNING | ABORTED | FINISHED_OK | ERROR
sandbox	no		Sandbox code. The operation will return only records for graphs from the specified sandbox.
graphId	no		Text Id, which is unique in a specified sandbox. The file path is relative to the sandbox root.
orderBy	no		Attribute for list ordering. Possible values: id | graphId | status | startTime | stopTime. By default, there is no ordering.
orderDescend	no	true	A 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 a 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|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. A 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 25.6. Parameters of suspend

parameter name	mandatory	default	description
sandbox	no	-	Text ID of a sandbox to suspend. If not specified, it suspends the 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 25.7. Parameters of resume

parameter name	mandatory	default	description
sandbox	no	-	Text Id of a sandbox to resume. If not specified, the server will be resumed.
verbose	no	MESSAGE	MESSAGE | FULL - how verbose should the possible error message be.

returns

Result message

Operation sandbox_create

This operation creates a specified sandbox. If it is sandbox of "partitioned" or "local" type, it also creates locations by "sandbox_add_location" operation.

parameters

Table 25.8. 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 a location to the specified sandbox. Can be only used with partitioned or local sandboxes.

parameters

Table 25.9. 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 a location from the specified sandbox. Only sandboxes of type partitioned or local can have locations asociated.

parameters

Table 25.10. 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 a specified sandbox as a ZIP archive.

parameters

Table 25.11. 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 a specified sandbox.

parameters

Table 25.12. 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 the 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 a current server configuration in XML format.

parameters

Table 25.13. 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 25.14. 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 a simple message, FULL for a 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 an application server configuration to make JMX Beans accessible.

This section describes how to configure a 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. 230)
- How to Configure JMX on Glassfish (p. 231)
- How to Configure JMX on WebSphere (p. 231)
- Possible Problems (p. 232)

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

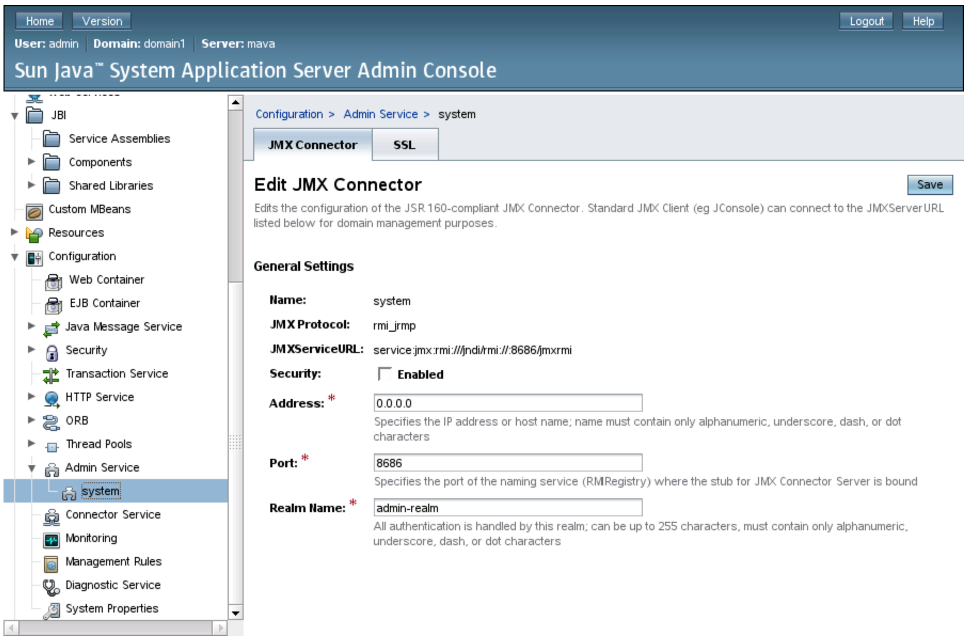
```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

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:



Edit JMX Connector

Edits the configuration of the JSR 160-compliant JMX Connector. Standard JMX Client (eg JConsole) can connect to the JMXServerURL listed below for domain management purposes.

General Settings

Name:	system
JMX Protocol:	rmi_jrmp
JMXServiceURL:	service:jmx:rmi:///jndi/rmi://:8686/jmxrmi
Security:	☐ Enabled
Address: *	0.0.0.0 Specifies the IP address or host name; name must contain only alphanumeric, underscore, dash, or dot characters
Port: *	8686 Specifies the port of the naming service (RMIRegistry) where the stub for JMX Connector Server is bound
Realm Name: *	admin-realm All authentication is handled by this realm; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

Figure 25.1. Glassfish JMX connector

With these values, you can use URL

```
service:jmx:rmi:///jndi/rmi://localhost:8686/jmxrmi
```

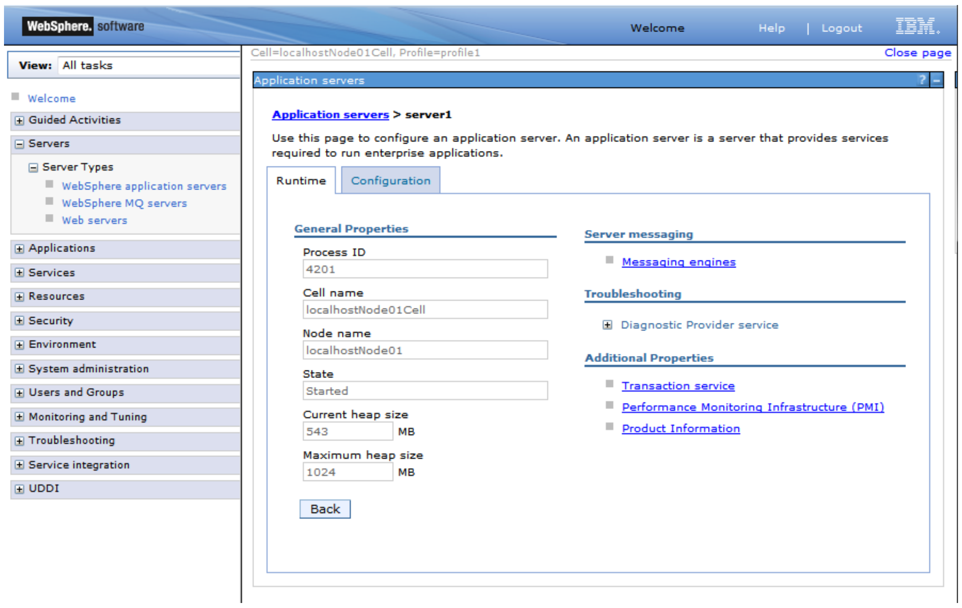
for connection to JMX server of JVM.

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]
```



Cell=localhostNode01Cell, Profile=profile1

Application servers > server1

Use this page to configure an application server. An application server is a server that provides services required to run enterprise applications.

General Properties

Process ID: 4201

Cell name: localhostNode01Cell

Node name: localhostNode01

State: Started

Current heap size: 543 MB

Maximum heap size: 1024 MB

Figure 25.2. WebSphere configuration

URL for connecting to JMX server is:

```
service:jmx:iiop://[host]:[port]/jndi/JMXConnector
```

where *host* is the host name you are connecting 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 a 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, complex lookups, etc.) for other application or directly to users.

Launch Services Overview

The architecture of a Launch Service is layered. It follows the basic design of multi-tiered applications utilizing a web browser.

Launch services let you build a user-friendly form that the user fills in and sends to the CloverETL Server for processing.

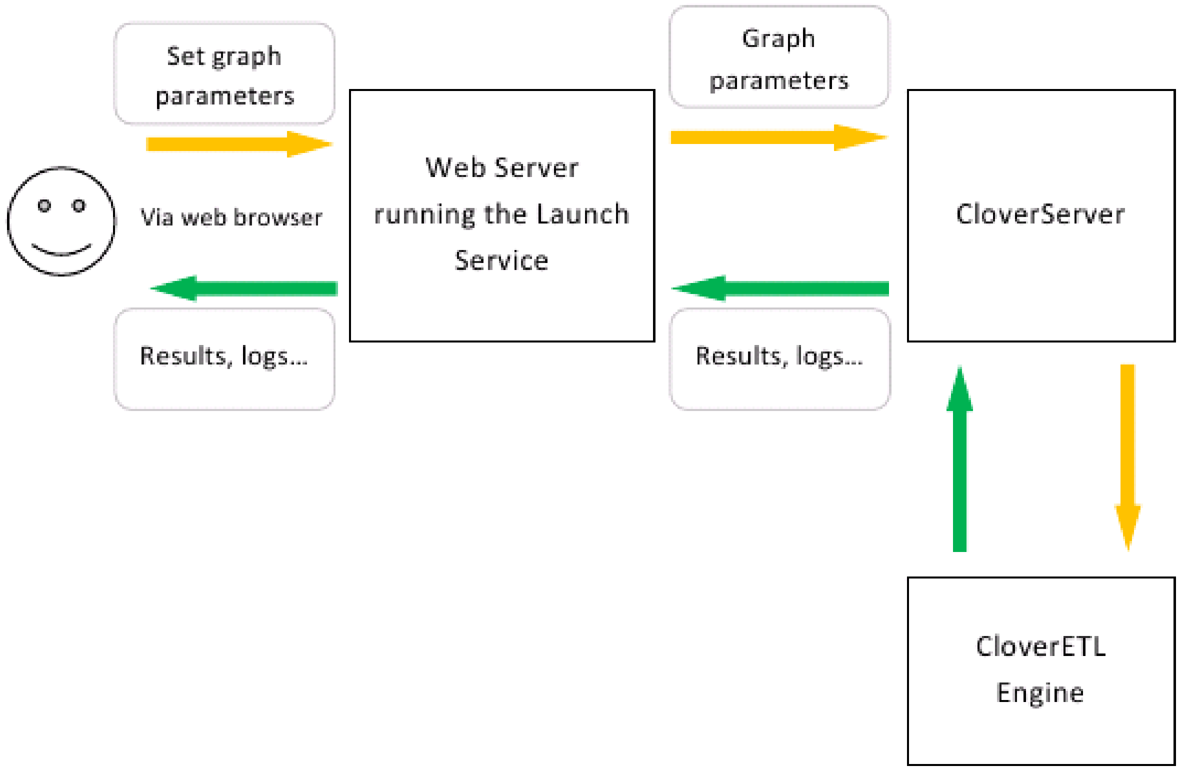


Figure 25.3. Launch Services and CloverETL Server as web application back-end

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 – 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 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.

Passing Files to Launch Sevices

If Launch service is designed to pass an input file to a graph or jobflow, the input dictionary entry has to be of type `readable.channel`.

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.

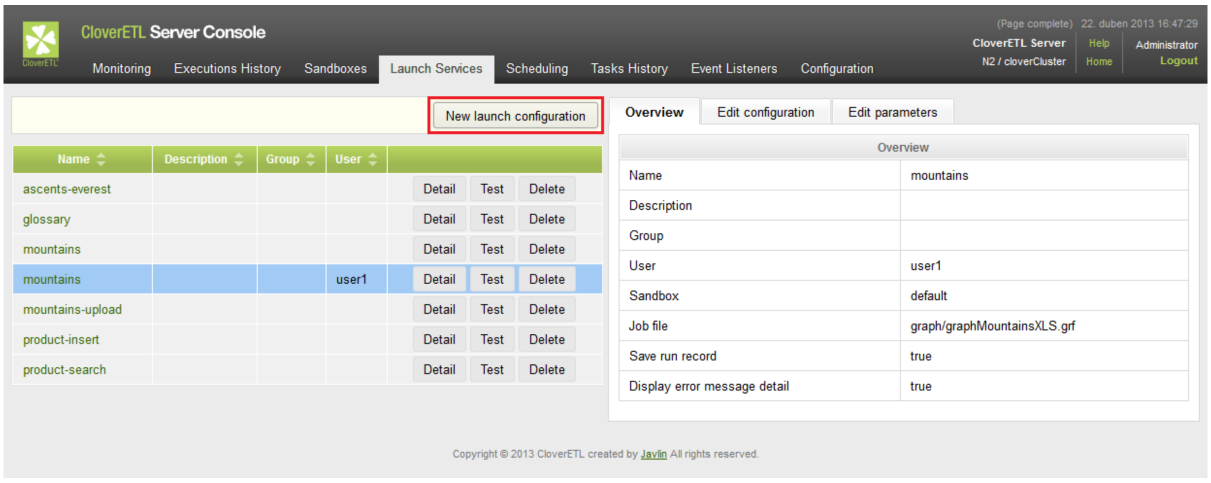
User restrictions can then be used to launch different jobs for different users, even though they use the same launch configuration (i.e. name). For example, developers may want to use a debug version of the job, while the end customers will want to use a production job. The user restriction can also 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.

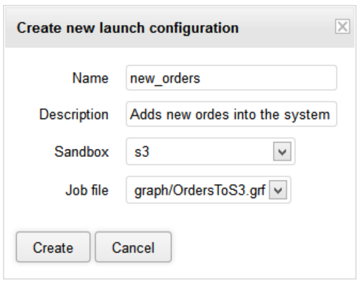


Name	Description	Group	User			
ascents-everest				Detail	Test	Delete
glossary				Detail	Test	Delete
mountains				Detail	Test	Delete
mountains			user1	Detail	Test	Delete
mountains-upload				Detail	Test	Delete
product-insert				Detail	Test	Delete
product-search				Detail	Test	Delete

Overview	
Name	mountains
Description	
Group	
User	user1
Sandbox	default
Job file	graph/graphMountainsXLS.grf
Save run record	true
Display error message detail	true

Figure 25.4. 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.



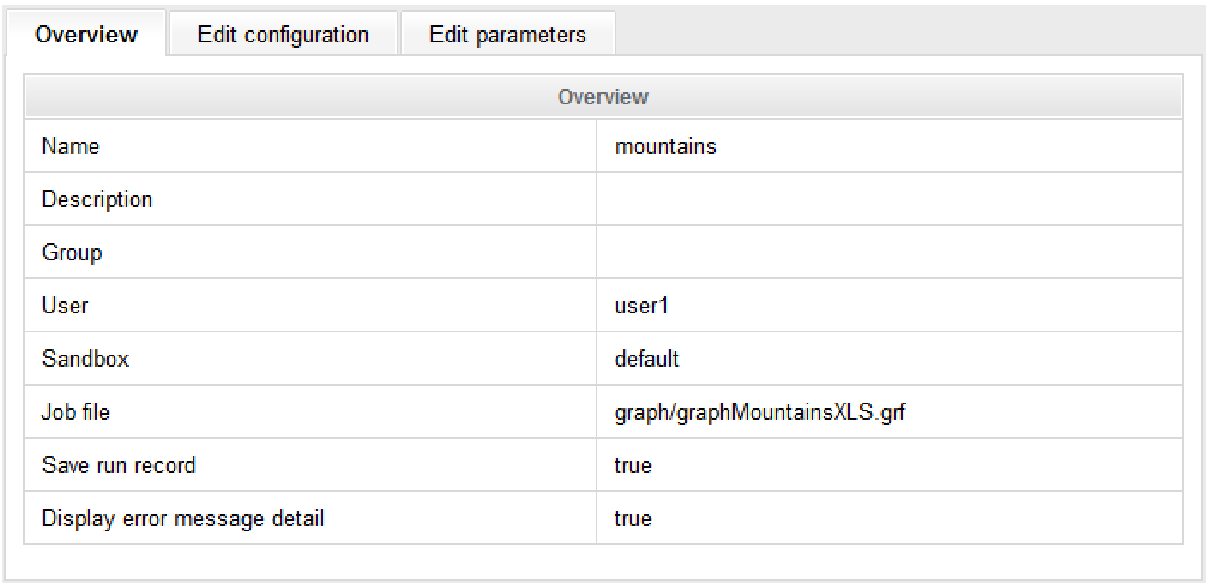
Create new launch configuration

Name	new_orders
Description	Adds new ordes into the system
Sandbox	s3
Job file	graph/OrdersToS3.grf

Figure 25.5. 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)

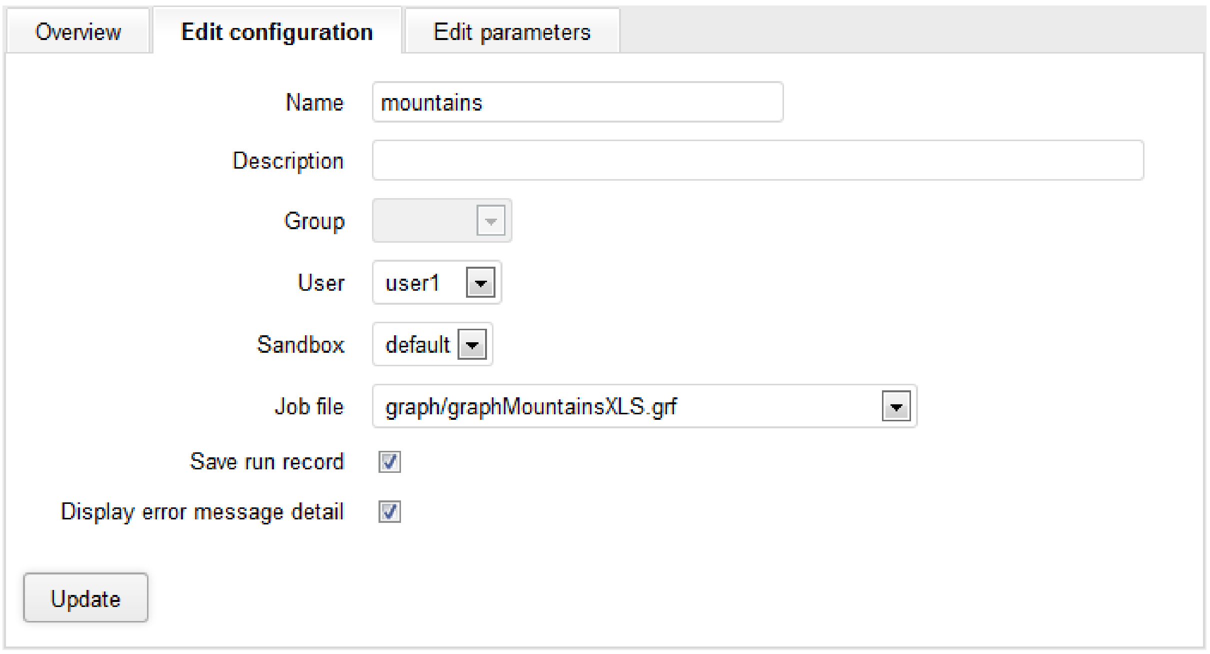


Overview	
Name	mountains
Description	
Group	
User	user1
Sandbox	default
Job file	graph/graphMountainsXLS.grf
Save run record	true
Display error message detail	true

Figure 25.6. Overview tab

The Overview tab shows the basic details about the launch configuration. These can be modified in the Edit Configuration tab:

Edit Configuration



Name: mountains
Description:
Group:
User: user1
Sandbox: default
Job file: graph/graphMountainsXLS.grf
Save run record: ✓
Display error message detail: ✓

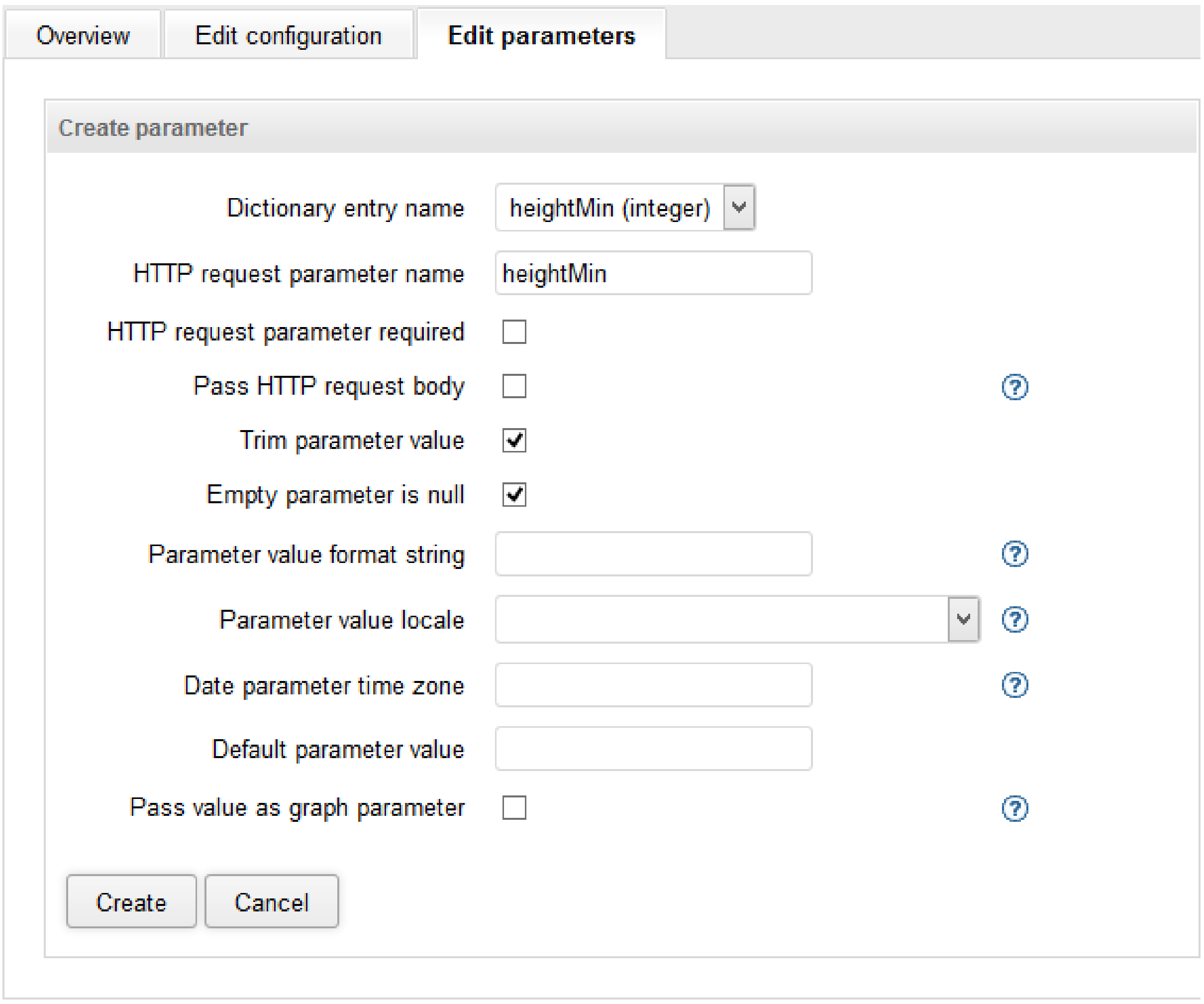
Figure 25.7. Edit Configuration tab

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

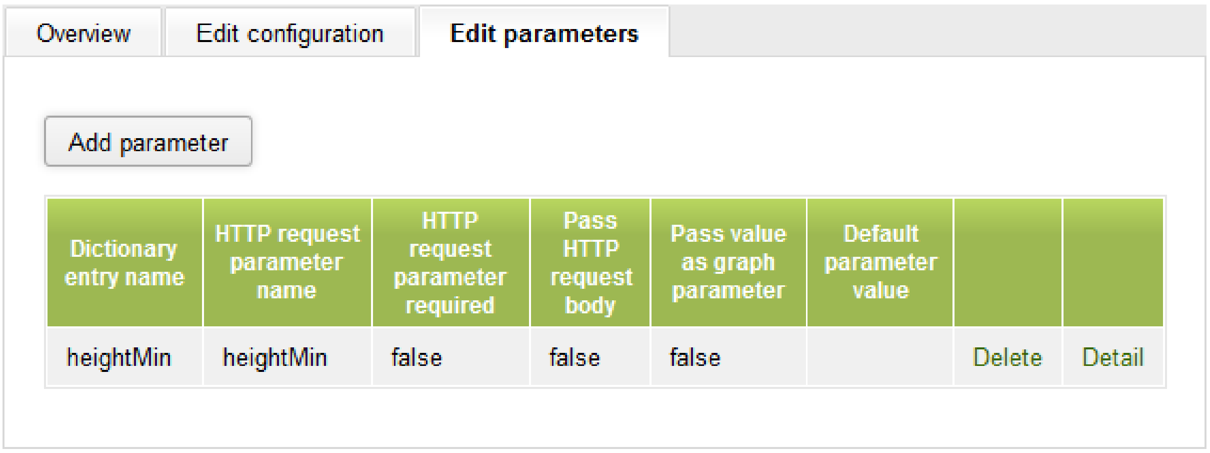
The “Edit parameters” tab can be used to configure parameter mappings for the launch configuration. The mappings are required for the Launch Service to be able to correctly assign parameters values based on the values sent in the launch request.



Dictionary entry name	heightMin (integer)
HTTP request parameter name	heightMin
HTTP request parameter required	☐
Pass HTTP request body	☐
Trim parameter value	☑
Empty parameter is null	☑
Parameter value format string	
Parameter value locale	
Date parameter time zone	
Default parameter value	
Pass value as graph parameter	☐

Figure 25.8. 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.



Dictionary entry name	HTTP request parameter name	HTTP request parameter required	Pass HTTP request body	Pass value as graph parameter	Default parameter value		
heightMin	heightMin	false	false	false		Delete	Detail

Figure 25.9. 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 using 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 a 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: 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 the 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.

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]
```

The HTTP protocol 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.

See also Groovy Code API permission (p. 128).

Variables Accessible in the Groovy Code

By default, these variables are accessible to the Groovy code

Table 25.15. Variables accessible in groovy code

type	key	description
com.cloveretl.server.config.CloverConfiguration	configuration	Provides configuration values for CloverETL Server.
javax.servlet.ServletContext	servletContext	Instance of ServletContext. Defines a set of methods that a servlet uses to communicate with its servlet container.
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

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: 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.*;

Response<List<Schedule>> list = serverFacade.findScheduleList(sessionToken, null);

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");

SchedulesList xmlList = new SchedulesList();
xmlList.setSchedulesList(list.getBean());

StringWriter writer = new StringWriter();
m.marshal(xmlList, writer);

return writer.toString();
```

Chapter 26. 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 27. 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 `engine.plugins.additional.src` config property.

See details about the possibilities with CloverETL configuration in *Part III, "Configuration" (p. 60)*

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 a WAR file. 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. 98) for details about overriding a server log4j configuration.

Chapter 28. 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.

SSL/TLS Issues

Java 7 and HTTPS

There may be issues when using Java 7 and HTTPS-based communication. There are servers that do not support TLS 1.0 (Transport Layer Security) protocol anymore and the Java 7 does not support newer versions of the TLS by default. To activate newer versions of the TLS protocol, user has to specify the system property `https.protocols` and set the value to `TLSv1,TLSv1.1,TLSv1.2`. Consult section on Production Server (p. 12) to find out how to set the system property on particular application server.

SSL-related Failures on WebLogic 12

Certain graphs using SSL-encrypted connections may fail on WebLogic 12 due to damaged library distributed with this application server. The issue can be identified by a SHA-1 digest error in the graph execution stacktrace:

```
...
Caused by: java.io.IOException: Could not convert socket to TLS
    at com.sun.mail.pop3.Protocol.stls(Protocol.java:659)
    at com.sun.mail.pop3.POP3Store.getPort(POP3Store.java:269)
    at com.sun.mail.pop3.POP3Store.protocolConnect(POP3Store.java:207)
Caused by: javax.net.ssl.SSLException: java.lang.SecurityException:
    SHA1 digest error for org/bouncycastle/jce/provider/JCEECPublicKey.class
...
```

To fix the issue, replace the library `[MW_HOME]/oracle_common/modules/bcprov-jdk16-1.45.jar` with the one downloaded directly from Bouncy Castle home page. Restart of the application server is required to load the new library.

Part VI. Cluster

Chapter 29. 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 accessible 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 itself cannot process a request. 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 only available 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 an arbitrary cluster node and moreover each instance contains only convenient components.

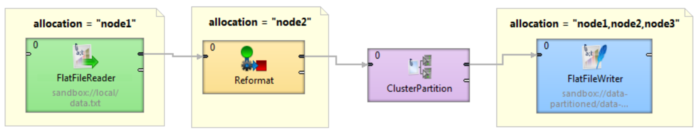


Figure 29.1. Component allocations example

This figure shows a sample graph with a few components with various component allocations. First component **FlatFileReader** requests to be executed on node1, the following Reformat component should be running on cluster node2, the ParallelPartition component is a special component which makes possible to change cardinality of allocation of two interconnected components (detailed description of cluster partitioning and gathering follows this section). **FlatFileWriter**, the last component, 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 a different cluster node (which is not necessary, even multiple workers can be associated with a single node). Worker on cluster node1 contains only FlatFileReader and first of three instances of FlatFileWriter component. Both components are connected by remote edges with components, which are running on node2. The worker running on node3 contains only FlatFileWriter fed by data remotely transferred from ParallelPartitioner running on node2.

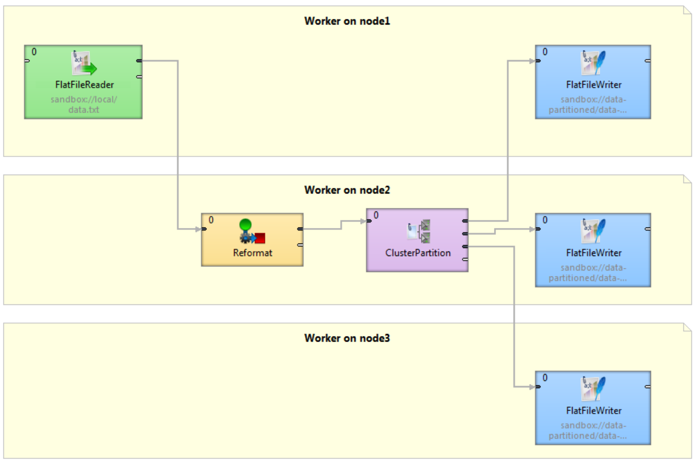
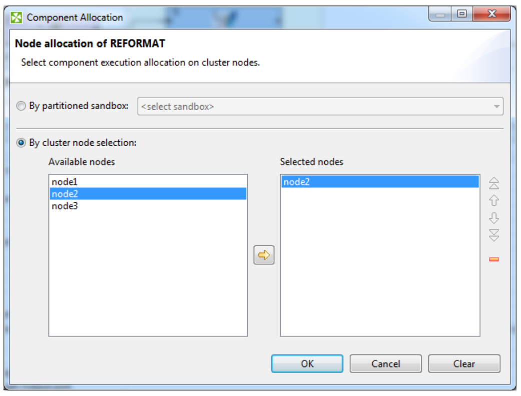


Figure 29.2. Graph decomposition based on component allocations

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.



Component Allocation

Node allocation of REFORMAT

Select component execution allocation on cluster nodes.

○ By partitioned sandbox: <select sandbox>

● By cluster node selection:

Available nodes	Selected nodes
node1	node2
node2	
node3	

Figure 29.3. 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. 250) for details.

  - allocation defined by a list of cluster node identifiers (a single cluster node can be used more times)

- **Reference to a partitioned sandbox** FlatFileReader, FlatFileWriter 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: ParallelPartition, ParallelLoadBalancingPartition, ParallelSimpleCopy, ParallelSimpleGather, ParallelMerge and ParallelRepartition. All the components 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.

ParallelPartition and **ParallelLoadBalancingPartition** work similar to a common partitioner, they change the data allocation from 1 to N. Component preceding the ParallelPartitioner run on just one node, whereas component behind the ParallelPartitioner run in parallel according to node allocation. **ParallelSimpleCopy** component can be use in similar locations. This component does not distribute the data records, but copies them to all output workers.

ParallelGather and **ParallelMerge** 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 ParallelPartitioner 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 have its 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 ParallelGather and ParallelMerge must have cardinality 1.* So it may be of any allocation, but the cardinality must be just 1.

- *Node allocation of components in front of the ParallelPartition, ParallelLoadBalancingPartition and ParallelSimpleCopy must have cardinality 1.*

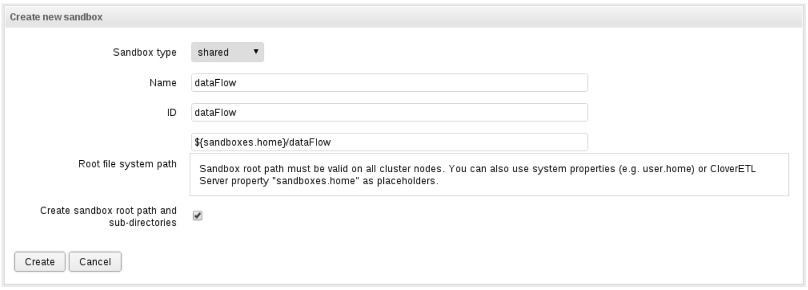
Partitioned and Local Sandboxes

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.


Create new sandbox

Sandbox type	shared
Name	dataFlow
ID	dataFlow
Root file system path	${sandboxes.home}/dataFlow
	Sandbox root path must be valid on all cluster nodes. You can also use system properties (e.g. user.home) or CloverETL Server property "sandboxes.home" as placeholders.
Create sandbox root path and sub-directories	✓

Figure 29.4. 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 intended 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:

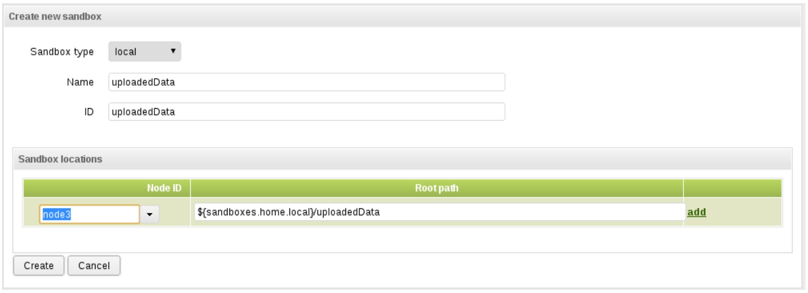
- `USERPROFILE` environment variable on Windows OS
- `user.home` system property (user home directory)
- `user.dir` system property (JVM process working directory)
- `java.io.tmpdir` system property (JVM process temporary directory)

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/writable 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. 252) 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.


Create new sandbox

Sandbox type	local
Name	uploadedData
ID	uploadedData

Sandbox locations

Node ID	Root path	
node3	${sandboxes.home.local}/uploadedData	add

Figure 29.5. Dialog form for creating a new local sandbox

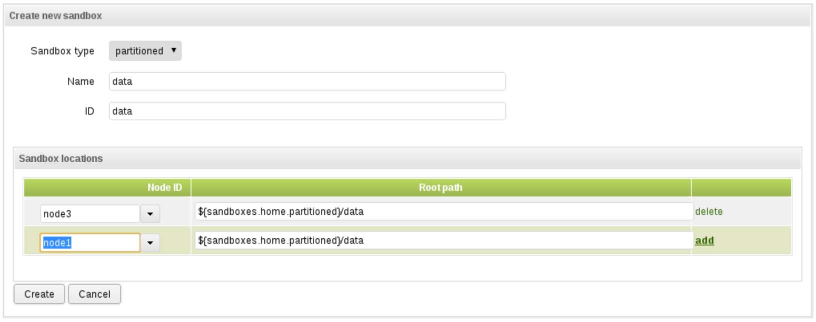
Sandbox location path is pre-filled with `sandboxes.home.local` placeholder which by default points to the `[user.data.home]/CloverETL/sandboxes-local`. The placeholder can be configured as any other CloverETL configuration property.

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.


Create new sandbox

Sandbox type	partitioned
Name	data
ID	data

Sandbox locations

Node ID	Root path	
node3	${sandboxes.home.partitioned}/data	delete
node1	${sandboxes.home.partitioned}/data	add

Figure 29.6. 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 `sandboxes.home.partitioned` placeholder which by default points to the `[user.data.home]/CloverETL/sandboxes-paritioned`. Anyway the `sandboxes.home.partitioned` config property may be configured as any other CloverETL Server configuration property. Note that directory must be readable/writable 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 the 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.

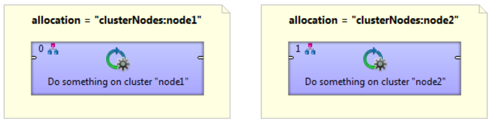
CloverETL Server evaluates the sandbox URL on each worker and provides an open stream to a local resource to the component.

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

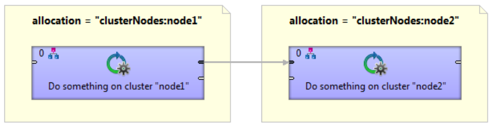
Basic component allocation

This example shows two component graph, where allocation ensures that the first component will be executed on cluster node1 and the second component will be executed on cluster node2.


allocation = "clusterNodes:node1"
Do something on cluster "node1"
allocation = "clusterNodes:node2"
Do something on cluster "node2"

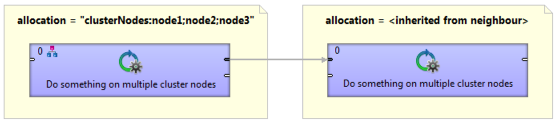
Basic component allocation with remote data transfer

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.


allocation = "clusterNodes:node1"
Do something on cluster "node1"
allocation = "clusterNodes:node2"
Do something on cluster "node2"

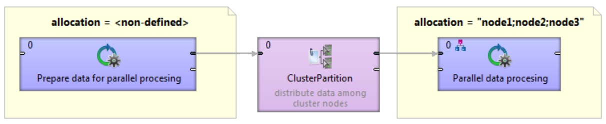
Multiple execution

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.


allocation = "clusterNodes:node1;node2;node3"
Do something on multiple cluster nodes
allocation = <inherited from neighbour>
Do something on multiple cluster nodes

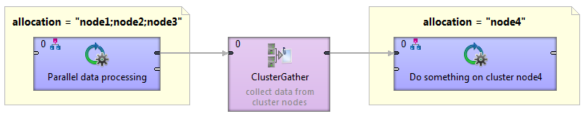
Cluster data partitioning

Graph with two allocations. First component has a single node allocation, which is not specified and is automatically derived to ensure minimal number of remote edges. The ParallelPartition component distribute records for further data processing on cluster node1, node2 and node3.


allocation = <non-defined>
Prepare data for parallel procesing
ClusterPartition
distribute data among cluster nodes
allocation = "node1;node2;node3"
Parallel data procesing

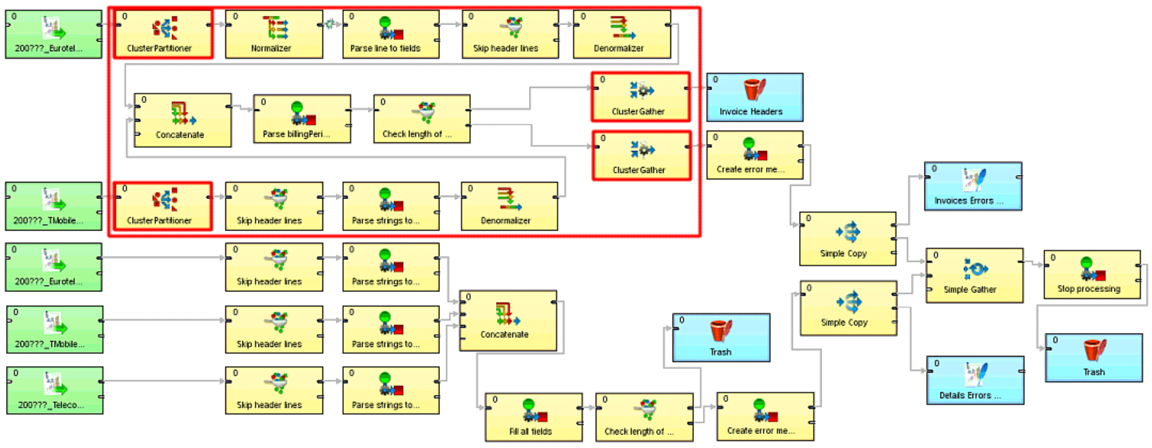
Cluster data gathering

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.


allocation = "node1;node2;node3"
Parallel data processing
ClusterGather
collect data from cluster nodes
allocation = "node4"
Do something on cluster node4

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

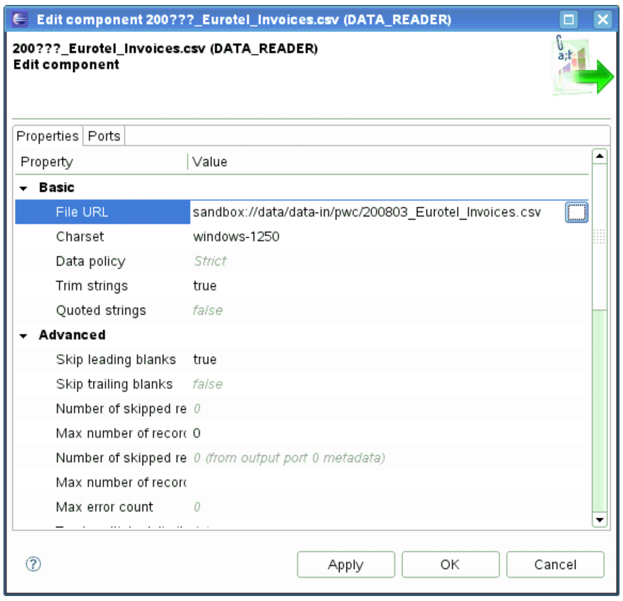
Please note there are four cluster components in the graph and these components define a point of change "node allocation", so the part of the graph demarcated by these components is highlighted by the red rectangle. Allocation of these component should be performed in parallel. This means that the components 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.

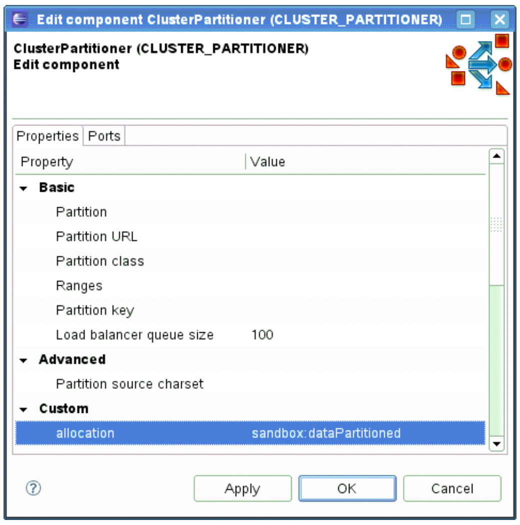


200???_Eurotel_Invoices.csv (DATA_READER)
Edit component

Property	Value
Basic	
File URL	sandbox://data/data-in/pwc/200803_Eurotel_Invoices.csv
Charset	windows-1250
Data policy	*Strict*
Trim strings	true
Quoted strings	*false*
Advanced	
Skip leading blanks	true
Skip trailing blanks	*false*
Number of skipped re	*0*
Max number of recor	0
Number of skipped re	*0 (from output port 0 metadata)*
Max number of recor	
Max error count	*0*

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.



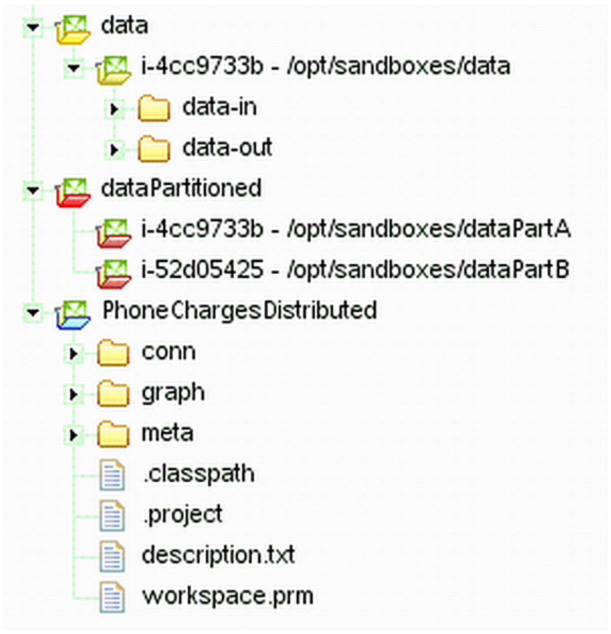
ClusterPartitioner (CLUSTER_PARTITIONER)
Edit component

Property	Value
Basic	
Partition	
Partition URL	
Partition class	
Ranges	
Partition key	
Load balancer queue size	100
Advanced	
Partition source charset	
Custom	
allocation	sandbox:dataPartitioned

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



```
data
    i-4cc9733b - /opt/sandboxes/data
        data-in
        data-out
dataPartitioned
    i-4cc9733b - /opt/sandboxes/dataPartA
    i-52d05425 - /opt/sandboxes/dataPartB
PhoneChargesDistributed
    conn
    graph
    meta
    .classpath
    .project
    description.txt
    workspace.prm
```

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:

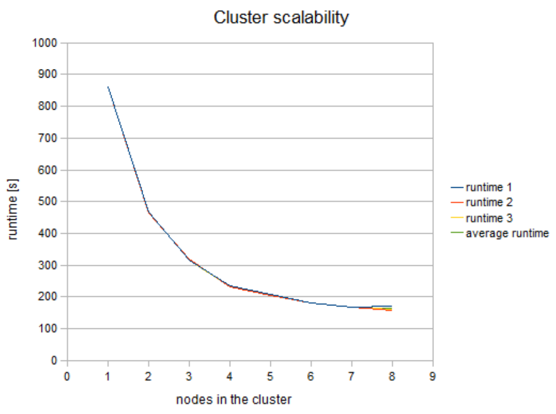


Figure 29.7. Cluster Scalability

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.

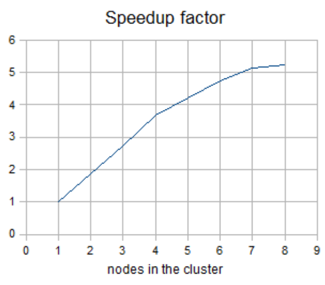


Figure 29.8. Speedup factor

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 30. 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 (p. 261)
- Optional Properties (p. 263)
- Example of 2 Node Cluster Configuration (p. 266)
- Jobs Load Balancing Properties (p. 269)

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. 265) for details.

Mandatory properties - these properties must be properly set on each node of the cluster

cluster.enabled	Switch whether the server shall start in the standalone or cluster node mode. The property isn't set at all (empty value) by default, which means, that the mode is chosen according to the loaded license. It's strongly recommended to set the property to "true" if also the other cluster properties are configured. Thus the cluster node will be initialized regardless the license. type: boolean
cluster.node.id	Each cluster node must have unique ID. type: String default: node01
cluster.jgroups.bind_address	IP address of ethernet interface, which is used for communication with another cluster nodes. Necessary for inter-node messaging. type: String, IP address default: 127.0.0.1
cluster.jgroups.start_port	Port where jGroups server listens for inter-node messages. type: int, port default: 7800
cluster.http.url	URL of the CloverETL cluster node. It must be HTTP/HTTPS URL to the root of a web application, thus typically 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. type: String, URL default: http://localhost:8080/clover

Following property must be set only when the node uses "remote" DB datasource (See property cluster.datasource.type (p. 265) 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 necessary to configure them explicitly.

Mandatory property for remote DB datasource access

cluster.jgroups.tcpping.initial_hosts	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 necessary to list all nodes of the cluster, but at least one of listed host:port must be running. Necessary for inter-node messaging.

type: String, in format: "IPaddress1[port1],IPaddress2[port2]"

default: 127.0.0.1[7800]

Optional Properties

Optional properties - these properties aren't vital for cluster configuration - default values are sufficient

cluster.jgroups.external_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.

type: String, IP address

cluster.jgroups.external_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.

type: int, port

cluster.jgroups.protocol.NAKACK.gc_lag

Number of delivered messages kept in the "sent messages buffer" of each jGroups view member. Messages are kept in sender cache even though they were reported as delivered by existing view members, because there may be some other member temporarily not in the view. The higher the number, the higher chance of reliable messages delivery in unreliable network environment. However the messages consume memory: aproximatelly 4kB for each message.

type: int

default: 10000

cluster.jgroups.protocol.NAKACK.xmit_table_obsolete_member_timeout

How long we keep obsolete member in the xmit-table. Time in millis. It's neccessary to recognition of member temporarily unaccessible and removed from the view. With previous NAKACK implementation, the member removed from the view was also automatically removed from xmit-table, so it appeared as new member when it re-joined the view. With current modified implementation the member is kept in the xmit-table for configured interval longer, so when it re-joins the view, it's known member and undelivered messages may be re-delivered to it. Member in the xmit-table isn't consuming memory.

type: long

default: 3600000

cluster.jgroups.protocol.AUTH.value

String used by jgroups member to authenticate to the group. Must be the same on all cluster nodes. Its protection against fake messages.

type: String

sandboxes.home.partitioned

This property is intended 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 unix-like systems it's typically /home/[username] type: String default: ${user.data.home}/CloverETL/sandboxes-partitioned
sandboxes.home.local	This property is intended 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 "user.data.home" configuration property which points to the home directory of the user which runs the JVM process. The directory location depends on the OS. On Unix-like systems it's typically /home/[username] type: String default: ${user.data.home}/CloverETL/sandboxes-local
cluster.shared_sandboxes_path	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. type: String
cluster.node.sendinfo.interval	time interval in ms; each node sends heart-beat with info about itself to another nodes; this interval specifies how often the info is sent under common circumstances type: int default: 2000
cluster.node.sendinfo.cluster.node.sendinfo.min_interval	time interval in ms; Specified minimum 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. type: int default: 500
cluster.node.sendinfo.history.interval	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 type: int default: 240000 (4 minutes)
cluster.node.remove.interval	time interval in ms; if no node info comes in this interval, the node is considered as lost and it is removed from the cluster type: int default: 50000
cluster.max_allowed_time_shift_between_nodes	

	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. type: int default: 2000
cluster.group.name	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. type: String defult: cloverCluster
cluster.jgroups.protocol.AUTH.value	Authentication string/password used for verification cluster nodes accessing the group. If this property is not specified, Cluster should be protected by firewall settings. type: String
cluster.datasource.type	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. type: String default: local
cluster.datasource.delegate.nodeIds	List of cluster node IDs (separated by comma ",") which this node may use as a 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 a 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. type: String

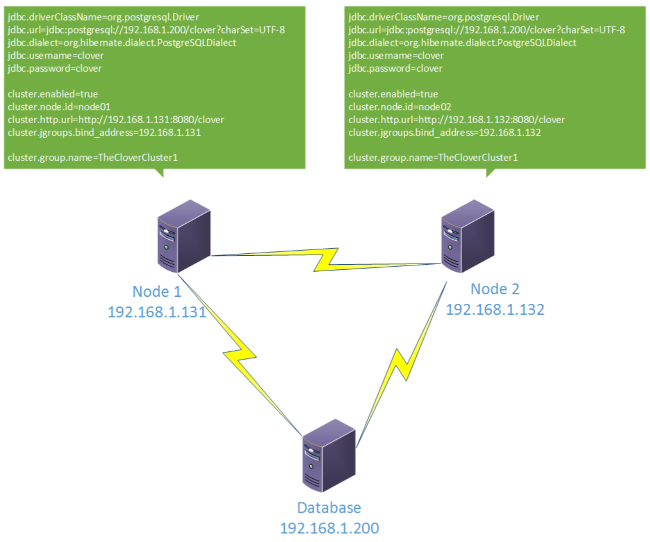
Example of 2 Node Cluster Configuration

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 necessary 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.



```
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
```

```
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
```

Figure 30.1. 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
```

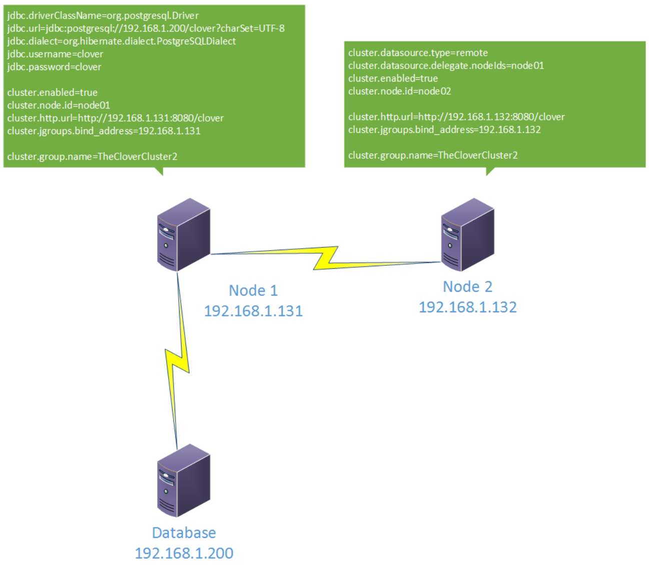
Configuration of node on 192.168.1.132

```
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.



```
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
```

```
cluster.datasource.type=remote
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
```

Figure 30.2. 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
```

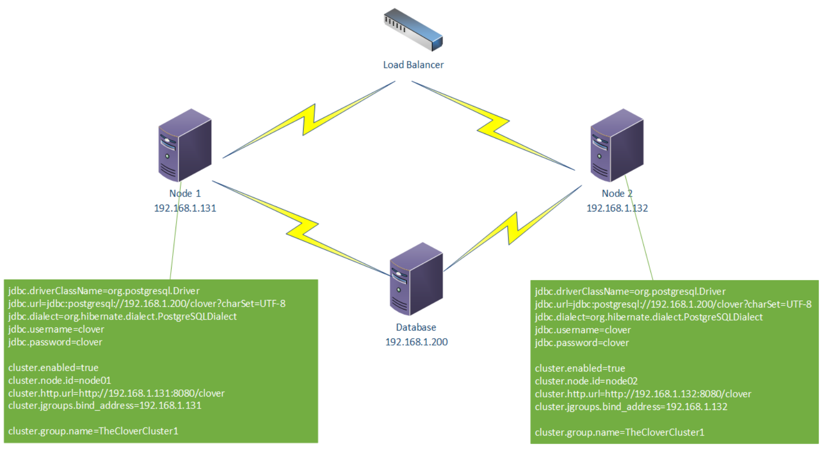
Configuration of node on 192.168.1.132

```
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 access 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.



```
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
```

```
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
```

Figure 30.3. 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 incoming 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 30.1. 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 memory 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 chosen by the loadbalancer. Value "1" means no bonus or penalty. "0" means that the node will be never chosen 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. 263).

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 Cannot Establish HTTP Connection to NodeB (p. 271)
- NodeA Cannot Establish TCP Connection (Port 7800 by Default) to NodeB (p. 271)
- NodeB is Killed or It Cannot Connect to the Database (p. 272)
- Auto-Resuming in Unreliable Network (p. 272)
- Long-Term Network Malfunction May Cause Jobs to Hang on (p. 273)

NodeA Cannot 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 cooperate 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"

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.check.checkMinInterval` - periodicity of cluster node checks (20000ms by default)
- `cluster.sync.connection.readTimeout` – HTTP connection response timeout (30000ms by default)
- `cluster.sync.connection.connectTimeout` – establishing HTTP connection timeout (7000ms by default)

NodeA Cannot 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 keeps waiting for the event from NodeB. 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”

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 (50000ms by default)

NodeB is Killed or It Cannot 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 considered as error.

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.touch.interval` – periodicity of database touch (20000ms by default)
- `cluster.node.touch.forced_stop.interval` – interval when the other nodes accept last touch (60000ms 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

Auto-Resuming in Unreliable Network

In version 4.4 we added auto-resuming of suspended nodes

Time-line describing the scenario:

- NodeB is suspended after connection loss
- **0s** NodeA successfully reestablishes connection to NodeB
- **120s** NodeA changes NodeB status to "forced_resume"
- NodeB attempts to resume itself if maximum auto-resume count is not reached.
- If the connection is lost again the cycle repeats, if maximum auto-resume count is exceeded the node will remain suspended until the counter is reset. To prevent suspend-resume cycles.
- **240m** auto-resume counter is reset

The following configuration properties serve to tune time intervals mentioned above:

- `cluster.node.check.intervalBeforeAutoresume` - time (in ms) the node has to be accessible to be forcibly resumed (120000 by default)
- `cluster.node.check.maxAutoresumeCount` - how many times a node may try to auto-resume itself (3 by default)
- `cluster.node.check.intervalResetAutoresumeCount=240 -` time (in minutes) before autoresume counter will be reset

Long-Term Network Malfunction May Cause Jobs to Hang on

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 31. 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`.

Chapter 32. Multiple CloverServer Instances on the same Host

Running multiple **CloverETL Server** instances on the same host is not recommended. If you do so, you should ensure that the instances do not interfere with each other.

- Each instance must run in a separate application server.
- Each instance must have its own port to listen. Application server can have some additional opened ports, therefore you might change the configuration of the application server as well.
- The instances must have different `${java.io.tmpdir}` directory.
- Each instance needs separate `${sanboxes.home}` directory.

See the documentation of particular application server for further information on running multiple instances of the application server on the same computer.

List of Figures

3.1. `setenv.sh` edited in Linux. 10
3.2. `setenv.bat` edited in Windows. 10
3.3. Apache Tomcat welcome page 14
3.4. Jetty welcome page 20
3.5. Adjusting Maximum heap size limit 24
3.6. Glassfish welcome page 28
3.7. JBoss AS welcome page 31
3.8. JBoss EAP welcome page 36
3.9. WebLogic welcome page 42
3.10. Login page of CloverETL Server without license 45
3.11. Add new license form 46
3.12. Update license form 47
3.13. CloverETL Server as the only running application on IBM WebSphere 52
12.1. Configured temp spaces overview - one default temp space on each cluster node 101
12.2. Newly added global temp space. 103
12.3. Temp spaces using environment variables and system properties 104
12.4. Disable operation reports action performed 105
12.5. Remove operation asks for confirmation in case there are data present in the temp space 106
13.1. Master password initialization 107
13.2. Graph parameters tab with initialized master password 107
14.1. Web GUI - section "Users" under "Configuration" 114
14.2. Web GUI - edit user 115
14.3. Web GUI - change password 115
14.4. Web GUI - groups assignment 116
14.5. Web GUI - section "Groups" 117
14.6. Web GUI - users assignment 118
14.7. Tree of permissions 118
15.1. Sandboxes Section in CloverETL Server Web GUI 129
15.2. Sandbox Permissions in CloverETL Server Web GUI 132
15.3. Web GUI - section "Sandboxes" - context menu on sandbox 133
15.4. Web GUI - section "Sandboxes" - context menu on folder 133
15.5. Web GUI - download sandbox as ZIP 134
15.6. Web GUI - upload ZIP to sandbox 134
15.7. Web GUI - upload ZIP results 135
15.8. Web GUI - download file as ZIP 135
15.9. Job config properties 139
16.1. Standalone server detail 142
16.2. Performance 143
16.3. Resource Utilization 143
16.4. CPU Load 143
16.5. Running jobs 144
16.6. System 144
16.7. Status History 144
16.8. Users' Accesses panel 144
16.9. Classloader cache 145
16.10. Status 145
16.11. Heartbeat 145
16.12. Threads 146
16.13. Quartz 146
16.14. Cluster overview 147
16.15. Node detail 148
16.16. Server Logs 149
17.1. Server Configuration Export screen 151
17.2. Server Configuration Import screen 152
17.3. Server Configuration uploaded 153

17.4. Outcome of the import preview for configuration from Example 17.1 154
17.5. Outcome of import preview for configuration after fixing by removal of broken group reference. 155
20.1. Web GUI - send e-mail 164
20.2. Web GUI - shell command 167
20.3. Web GUI - Graph execution task 170
20.4. Web GUI - Graph execution task 171
20.5. Web GUI - Jobflow execution task 172
20.6. Web GUI - "Abort job" 174
20.7. Web GUI - archive records 176
20.8. Web GUI - Task JMS message editor 178
21.1. Web GUI - "Manual task execution" form 182
22.1. Web GUI - section "Scheduling" - create new 183
22.2. Web GUI - onetime schedule form 184
22.3. Web GUI - schedule form - calendar 185
22.4. Web GUI - periodical schedule form 186
22.5. Cron periodical schedule form 187
22.6. Editing the cron expression - minutes 187
22.7. Editing the cron expression - days of week 188
22.8. Schedule allocation - Any node 189
22.9. Schedule allocation - One ore more specific nodes 189
23.1. Executions History - executions table 191
23.2. Executions History - overall perspective 193
23.3. Executions Hierarchy with docked list of jobs 193
24.1. Listeners 194
24.2. Web GUI - graph timeout event 196
24.3. Event source graph isn't specified, thus listener works for all graphs in specified sandbox 198
24.4. Web GUI - e-mail notification about graph failure 199
24.5. Web GUI - email notification about graph success 200
24.6. Web GUI - backup of data processed by graph 201
24.7. Web GUI - jobflow timeout event 203
24.8. Web GUI - creating a File Event listener 212
24.9. Web GUI - creating a Task Failure listener 218
25.1. Glassfish JMX connector 231
25.2. WebSphere configuration 232
25.3. Launch Services and CloverETL Server as web application back-end 234
25.4. Launch Services section 235
25.5. Creating a new launch configuration 236
25.6. Overview tab 236
25.7. Edit Configuration tab 237
25.8. Creating new parameter 238
25.9. Edit Parameters tab 238
29.1. Component allocations example 248
29.2. Graph decomposition based on component allocations 249
29.3. Component allocation dialog 249
29.4. Dialog form for creating new shared sandbox 251
29.5. Dialog form for creating a new local sandbox 251
29.6. Dialog form for creating new local sandbox 252
29.7. Cluster Scalability 258
29.8. Speedup factor 258
30.1. Configuration of 2-nodes cluster, each node has access to database 266
30.2. Configuration of 2-nodes cluster, one node without direct access to database 267
30.3. Configuration of 2-nodes cluster, one node without direct access to database 268

List of Tables

1.1. CloverETL Server and CloverETL Engine comparison 3
2.1. Hardware requirements of CloverETL Server 5
2.2. CloverETL Server Compatibility Matrix 6
4.1. JVM Memory Structure 55
4.2. Firewall Exceptions 57
9.1. General configuration 88
9.2. Defaults for job execution configuration - see Job Config Properties for details 93
10.1. Parameters 95
13.1. Secure parameters configuration parameters 108
14.1. After default installation on an empty DB, admin user is created automatically 114
14.2. User attributes 114
14.3. Default groups created during installation 117
15.1. Sandbox attributes 130
15.2. Sandbox permissions 132
15.3. ZIP upload parameters 135
15.4. Job config parameters 137
19.1. Defaults for graph execution configuration - see section Graph config properties for details 158
19.2. passed parameters 159
19.3. passed parameters 160
19.4. passed parameters 160
20.1. Attributes of "Send e-mail" task 163
20.2. Placeholders useful in e-mail templates 166
20.3. Attributes of "Execute shell command" task 167
20.4. Parameters of "Execute shell command" task 168
20.5. Parameters of "Execute shell command" task - available in scheduling 168
20.6. Parameters of "Execute shell command" task - available in listeners 168
20.7. Parameters of "Execute shell command" task - available in manual task execution 169
20.8. Attributes of "Graph execution" task 170
20.9. Additional parameters available in Event Listeners 171
20.10. Attributes of "Jobflow execution" task 172
20.11. Attributes of "Abort job" task 174
20.12. Attributes of "Archivator" task 175
20.13. Attributes of JMS message task 177
20.14. Parameters of "Send a JMS Message" 178
20.15. List of variables available in Groovy code 180
22.1. Onetime schedule attributes 184
22.2. Periodical schedule attributes 186
22.3. Cron periodical schedule attributes 187
23.1. Persistent run record attributes 192
24.1. Attributes of JMS message task 205
24.2. Variables accessible in groovy code 207
24.3. Properties Elements 208
24.4. "data" elements 208
24.5. Attributes of Universal message task 210
24.6. Variables accessible in Groovy code 211
24.7. Parameters usable in task failure e-mail template 219
24.8. File Event Listener specific parameters usable in task failure e-mail template 219
25.1. Parameters of graph_run 221
25.2. Parameters of graph_status 222
25.3. Parameters of graph_kill 222
25.4. Parameters of sandbox_content 223
25.5. Parameters of executions_history 224
25.6. Parameters of suspend 225
25.7. Parameters of resume 225
25.8. Parameters of sandbox create 226

25.9. Parameters of sandbox add location 226
25.10. Parameters of sandbox add location 227
25.11. Parameters 227
25.12. Parameters 227
25.13. Parameters of server configuration export 228
25.14. Parameters of server configuration import 229
25.15. Variables accessible in groovy code 241
30.1. Load balancing properties 269

List of Examples

17.1. Example of simple configuration defining one new server user. 153