Copyright and disclaimer

Copyright © 2003, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. UNIX is a registered trademark of The Open Group.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Rosette® Linguistics Platform Copyright © 2000-2011 Basis Technology Corp. All rights reserved.

Teragram Language Identification Software Copyright © 1997-2005 Teragram Corporation. All rights reserved.
# Table of Contents

1. What is CloverETL Server ............................................................................................................. 1
2. Installation ......................................................................................................................................... 2
   - Apache Tomcat ............................................................................................................................... 2
   - Jetty ............................................................................................................................................... 4
   - IBM Websphere .............................................................................................................................. 6
   - Glassfish / Sun Java System Application Server ........................................................................... 8
   - JBoss .............................................................................................................................................. 9
   - Possible issues during installation ................................................................................................. 11
3. Graphs on Server Side - Sandboxes ................................................................................................ 15
   - Referencing files from the graph ................................................................................................. 16
   - Sandbox Security and Permissions .............................................................................................. 17
   - Sandbox Content .......................................................................................................................... 17
   - Graph config properties .................................................................................................................. 21
4. Users and Groups ............................................................................................................................ 24
   - LDAP authentication ..................................................................................................................... 24
   - Web GUI section Users .................................................................................................................. 26
   - Web GUI section Groups ............................................................................................................... 28
5. Scheduling ....................................................................................................................................... 31
   - Timetable Setting .......................................................................................................................... 31
   - Tasks .......................................................................................................................................... 34
6. Graph Event Listeners ..................................................................................................................... 40
   - Graph Events ............................................................................................................................... 40
   - Listener ........................................................................................................................................ 41
   - Tasks .......................................................................................................................................... 41
   - Use cases ..................................................................................................................................... 45
7. JMS messages listeners .................................................................................................................... 48
   - Optional Groovy code ................................................................................................................... 49
   - Message data available for further processing ............................................................................ 50
8. Universal event listeners .................................................................................................................. 52
   - Groovy code ............................................................................................................................... 52
9. Manual task execution ...................................................................................................................... 53
10. File event listeners .......................................................................................................................... 54
    - Observed file .............................................................................................................................. 54
    - File Events ................................................................................................................................. 55
    - Check interval, Task and Use cases ............................................................................................ 56
11. WebDAV ....................................................................................................................................... 57
    - WebDAV clients .......................................................................................................................... 57
    - WebDAV authentication/authorization ........................................................................................ 57
12. Simple HTTP API .......................................................................................................................... 59
    - Operation help ........................................................................................................................... 59
    - Operation graph_run .................................................................................................................. 59
    - Operation graph_status ............................................................................................................ 60
    - Operation graph_kill .................................................................................................................. 61
    - Operation server_jobs ................................................................................................................ 61
    - Operation sandbox_list .............................................................................................................. 61
    - Operation sandbox_content ........................................................................................................ 61
    - Operation executions_history ................................................................................................... 62
    - Operation suspend ..................................................................................................................... 63
    - Operation resume ....................................................................................................................... 63
13. JMX mBean ................................................................................................................................... 65
    - JMX configuration ....................................................................................................................... 65
    - Operations .................................................................................................................................. 69
14. SOAP WebService API ................................................................................................................. 73
    - SOAP WS Client ........................................................................................................................ 73
    - SOAP WS API authentication/authorization ............................................................................... 73
Chapter 1. What is CloverETL Server

CloverETL Server (CS) is the most recent member of CloverETL products family. It introduces the powerful Clover tool into the world of enterprise applications. CloverETL Server itself is an enterprise class application, thus it is shipped as WAR file (WAR stands for Web Archive). CS is tested and works on Apache Tomcat web container, Sun Glassfish application server or IBM Websphere application server. CloverETL Server is basically runtime environment for graphs, which brings new possibilities how to integrate Clover with your own software. Whereas CloverEngine can be integrated only as embedded library, CS implements several interfaces which can be called by another applications using common protocols like http. In addition, CS implements some optimizations of threads and memory management.

Table 1.1. CloverETL server and CloverETL engine comparison

<table>
<thead>
<tr>
<th></th>
<th>CloverETL Server</th>
<th>CloverEngine as executable tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>possibilities of executing graphs</td>
<td>by calling http (or JMX, etc.) APIs (See details in Chapter 12, Simple HTTP API (p. 59).)</td>
<td>by executing external process or by calling java API</td>
</tr>
<tr>
<td>engine initialization</td>
<td>during server startup</td>
<td>init is called for each graph execution</td>
</tr>
<tr>
<td>thread and memory optimization</td>
<td>threads recycling, graphs cache, etc.</td>
<td>not implemented</td>
</tr>
<tr>
<td>scheduling</td>
<td>scheduling by timetable, onetime trigger, logging included</td>
<td>external tools (i.e. Cron) can be used</td>
</tr>
<tr>
<td>statistics</td>
<td>each graph execution has its own log file and result status is stored; each event triggered by the CS is logged</td>
<td>not implemented</td>
</tr>
<tr>
<td>monitoring</td>
<td>If graph fails, event listener will be notified. It may send email, execute shell command or execute another graph. See details in Chapter 6, Graph Event Listeners (p. 40) Additionally server implements various APIs (HTTP and JMX) which may be used for monitoring of server/graphs status.</td>
<td>JMX mBean can be used while graph is running</td>
</tr>
<tr>
<td>storage of graphs and related files</td>
<td>graphs are stored on server file system in so called sandboxes</td>
<td></td>
</tr>
<tr>
<td>security and authorization support</td>
<td>CS supports users/groups management, so each sandbox may have its own access privileges set. All interfaces require authentication. See details in Chapter 3, Graphs on Server Side - Sandboxes (p. 15).</td>
<td>passwords entered by user may be encrypted</td>
</tr>
<tr>
<td>integration capabilities</td>
<td>CS provides APIs which can be called using common protocols like HTTP. See details in Chapter 12, Simple HTTP API (p. 59).</td>
<td>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.</td>
</tr>
<tr>
<td>development of graphs</td>
<td>CS supports team cooperation above one project (sandbox). CloverETL Designer will be integrated with CS in further versions.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2. Installation

CloverETL Server is shipped as a Web application archive (WAR file). Use standard methods for deploying a web application on your application server. Detailed information concerning the installation on a specific application server can be found in the chapters below.

The default installation (without changes to the configuration) does not need any extra database server. It uses the embedded Apache Derby DB. What is more, it does not need any subsequent configuration. CloverETL Server configures itself during the first startup. Database tables and some necessary records are automatically created on the first startup with an empty database. In the sandboxes section of the web GUI, you can check that there are sandboxes created with some demo graphs.

After successful installation open your browser and access the following URLs:

CloverETL web GUI - http://[host]:[port]/[contextPath]/gui

CloverETL HTTP API test page - http://[host]:[port]/[contextPath]/index.jsp

List of available installations:

- Apache Tomcat (p. 2)
- Jetty (p. 4)
- IBM Websphere (p. 6)
- Glassfish / Sun Java System Application Server (p. 8)

In case of problems during the installation see Possible issues during installation (p. 11).

Apache Tomcat

Installation of Apache Tomcat

CloverETL Server requires Apache Tomcat version 6.0.x to run.

If you have Apache Tomcat already installed, you can move on to the next section.

1. Download the binary distribution from http://tomcat.apache.org/download-60.cgi.
2. After you download the zip file, unpack it.
3. Run Tomcat by [tomcat_home]/bin/startup.sh (or [tomcat_home]/bin/tomcat.exe on Windows OS).
5. Apache Tomcat is installed.

If in need of detailed installation instructions, go to: http://tomcat.apache.org/tomcat-6.0-doc/setup.html

Installation of CloverETL Server

The installation is by default a very simple task:
1. Download the web archive file (clover.war) containing CloverETL Server for Apache Tomcat.

2. Check if prerequisites are met:

   • JDK or JRE version 1.6.x or higher
   
   • JAVA_HOME and JRE_HOME environment variables have to be set.
   
   • Apache Tomcat 6.0.x is installed. CloverETL Server is developed and tested with the Apache Tomcat 6.0.x container (it may work unpredictably with different versions). See Installation of Apache Tomcat (p. 2) for details.
   
   • It is strongly recommended to change default limits for the heap and "perm gen" memory spaces.

   You can set the minimum and maximum memory heap size by adjusting the "Xms" and "Xmx" JVM parameters. You can set JVM parameters for Tomcat by setting the environment variable JAVA_OPTS in the [TOMCAT_HOME]/bin/setenv.sh file (if it does not exist, you may create it). For instance, the minimum heap size being 128 MB and maximum heap size 1024 MB, type: JAVA_OPTS="-Xms128m -Xmx1024m". The best limits depend on many conditions, i.e. transformations which CloverETL should execute. If you have no idea about the memory required for the transformations, a maximum of 1 GB is recommended.

   You can set the maximum limit of "PermGen space" by the JVM parameter "-XX:MaxPermSize=256m". By default, it is just 64 MB which is not enough for enterprise applications. A suitable memory limit depends on various criteria, but 256 MB would make a good choice in most cases. If the PermGen space maximum is too low, "OutOfMemoryError: PermGen space" may occur.

   • For performance reasons, it is recommended the application is run in the "server" mode.

   Apache Tomcat does not run in the server mode by default. You can set the server mode by setting the "-server" JVM parameter. You can set the JVM parameter for Tomcat by setting the environment variable JAVA_OPTS in the [TOMCAT_HOME]/bin/setenv.sh file (if it does not exist, you may create it). That is: JAVA_OPTS="-server".

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

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

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

5. Check whether CloverETL Server is running on URLs:

   • Web-app root

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

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

     http://localhost:8080/clover/

   • web GUI

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

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

     http://localhost:8080/clover/gui
Chapter 2. Installation

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

**Installation of CloverETL Server License**

CloverETL Server requires a valid license for executing graphs. You can install CloverETL Server without any license, but no graph will be executed.

For Tomcat, the license is distributed as a separate web application. It is not necessary to install the license first. You can first install CloverETL Server and then its license.

1. Download the web archive file `clover-license.war`
2. Copy `clover-license.war` to the `[tomcat_home]/webapps` directory.
3. The war file should be detected and deployed automatically without restarting Tomcat.
4. Check whether the license web-app is in operation on URL:
   ```
   http://[host]:[port]/clover-license/ (contextPath “clover-license” is mandatory and cannot by changed)
   ```

   **Note**

   CloverETL license can be changed any time by re-deploying `clover-license.war`. Afterwards, you have to let CloverETL Server know the license has changed.

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

   **Warning**: Keep in mind that during the WAR file redeployment, directory `[tomcat_home]/webapps/[contextPath]` has to be deleted. If the Tomcat is running, it should do it automatically, but we suggest to check it manually, otherwise changes will not take any effect.

**Apache Tomcat on IBM AS/400 (iSeries)**

To run CloverETL Server on the iSeries platform, the requirements are:

1. Java 6.0 32-bit
2. Run java with parameter `-Djava.awt.headless=true`

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

```bash
JAVA_HOME=/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit
JAVA_OPTS="-Djava.awt.headless=true"
```

**Jetty**

**Installation of CloverETL Server**

1. Download the web archive file (`clover.war`) containing the CloverETL Server application which is built for Jetty.
Chapter 2. Installation

2. Check if prerequisites are met:
   - JDK or JRE version 1.6.x or higher
   - Jetty 6.1.x - only this particular version is supported
     All jetty-6 releases are available from http://jetty.codehaus.org/jetty/. As of Jetty 7, there have been huge differences in distribution packages as it is distributed by Eclipse foundation. Jetty 7 is not supported.

3. Copy clover.war to [JETTY_HOME]/webapps.

4. Create a context file clover.xml in [JETTY_HOME]/contexts and fill it with the following lines:

   ```xml
   <?xml version="1.0" encoding="ISO-8859-1"?>
   <!DOCTYPE Configure PUBLIC "+//Jetty//Configure//EN" "http://www.eclipse.org/jetty/configure.dtd">
   <Configure class="org.mortbay.jetty.webapp.WebAppContext">
     <Set name="contextPath">/clover</Set>
     <Set name="war"><SystemProperty name="jetty.home" default="."/>/webapps/clover.war</Set>
   </Configure>
   ```

   clover.xml will be detected by Jetty and the application will be loaded automatically.

5. Run [JETTY_HOME]/bin/jetty.sh start. (or [JETTY_HOME]/bin/Jetty-Service.exe on windows OS)

   You can check if the server is running e.g. on http://localhost:8080/test/.

**Installation of CloverETL Server license**

In order to execute graphs, CloverETL Server requires a valid license file. Despite that, you can install CloverETL Server without a license, but no graph will be executed.

1. Get the license.dat file.

2. Set the CloverETL Server license.file parameter to the path to license.dat.

   There are more ways how to achieve this. The most direct way is to create an environment or a system property called clover_license_file (see Chapter 16, Configuration (p. 81) for a description of all possibilities).

   If you are using Linux, follow these instructions:
   - Edit [JETTY_HOME]/bin/jetty.sh
   - Add a new line:

     ```bash
     export clover_license_file=[absolute_path_to_license_file]/license.dat
     ```
   - Restart Jetty.

   **Note**

   CloverETL license can be changed any time by replacing the license.dat file. Afterwards, you have to let CloverETL Server know the license has changed.
   - Go to server web GUI → Monitoring → License
Chapter 2. Installation

- Click **Reload license**.
- Alternatively, you can restart the CloverETL Server application.

---

**IBM Websphere**

**Installation of CloverETL Server**

1. get the web archive file (**clover.war**) with CloverETL Server application which is built for Websphere

2. check prerequisites
   - JDK or JRE version 1.6.x or higher
   - IBM Websphere 6.1 or IBM Websphere 7.0 (see [http://www.ibm.com/developerworks/downloads/ws/was/](http://www.ibm.com/developerworks/downloads/ws/was/))

3. deploy WAR file
   - go to **Integrated Solutions Console**
   - * go to section **Applications** → **New Application** → **New Enterprise Application**

4. change class loader setting
   CloverETL requires different class-loader settings in WebSphere 6 and WebSphere 7. In WebSphere 7, it is by default **Classes loaded with parent class loader first**, whereas in WebSphere 6, the default value must be changed to **Classes loaded with application class loader first**
   - * change **Applications** → **Clover** → **Manage Modules** → **clover.war** → **Class loader Order** → **Classes loaded with application class loader first**

5. configure system property on Websphere 6
   WebSphere 6 sets system property "javax.xml.transform.TransformerFactory" to the value "com.ibm.xtq.xslt.jaxp.compiler.TransformerFactoryImpl". This setting overrides default value used by CloverETL Server which is vital for webservice API. So please set explicitly correct value in the Websphere admin console.

   **Note**
   It is not necessary to set this in Websphere 7. You can skip the workaround below and advance to the next step.

   - * go to **Integrated Solutions Console**
   - * go to **Servers** → **Application servers** → [server1] (or the other server of correct name) → **Java and Process Management** → **Java Virtual Machine**
   - * In this section, add "-Djavax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl" to the input labelled "Generic JVM arguments"
   - * Then submit the form by "OK" button, then commit changes by "save" link. This change needs restart of Websphere to take effect.
Chapter 2. Installation

- You can check whether its properly is set or not in CloverETL Server web GUI, in the section "monitoring", "System properties" tab. There should be "javax.xml.transform.TransformerFactory" system property with correct value.

6. configure logging

WebSphere loggers do not use log4j by default which may cause CloverETL Server logging to be ill-configured. As a result, some CloverETL Engine messages are missing in graph execution logs. Thus it is recommended to configure WebSphere properly to use log4j. Add this config file to the WebSphere directory:

```
AppServer/
profiles/AppSrv01/properties/commons-logging.properties
```

Content of the file should be like this:

```
priority=1
org.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
```

Afterwards, copy Jar files from the clover.war/WEB-INF/lib archive to the AppServer/lib directory. Copy all files like commons-logging-*.jar and log4j-*.jar.

7. try if the server is running

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

```
http://localhost:9080/clover
```

Installation of CloverETL Server license

CloverETL Server requires a valid license for executing graphs. You can install CloverETL Server without license, but no graph will be executed.

1. get file license.dat

2. set CloverETL Server parameter license.file to the path to the license.dat file

There are more ways how to do this. The most direct way is to set environment property clover_license_file. (See Chapter 16, Configuration (p. 81) for description of all possibilities).

- go to Integrated Solutions Console

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


- create system property named clover_license_file whose value is the absolute path to license.dat on the file system

- Then you have to let CloverETL Server know the license has changed. Go to web GUI → Monitoring → License. Then click the Reload license button. Or you can restart CloverETL Server application.

**Note**

CloverETL license can be changed any time by replacing file license.dat. Then you have to let CloverETL Server know the license has changed.
Chapter 2. Installation

- Go to web GUI → monitoring section → license tab
- Then click the button reload license.
- Or you can restart CloverETL Server application.

Glassfish / Sun Java System Application Server

Installation of CloverETL Server

1. get web archive file (clover.war) with CloverETL Server application which is built for Glassfish (Tomcat)

2. check prerequisites
   - JDK or JRE version 1.6.x or higher
   - Glassfish (CloverETL Server is tested with V2.1, see http://glassfish.java.net/public/downloadsindex.html#top)

3. deploy WAR file

   Fill in attributes Application name and Context Root with value "clover". Fill in path to the WAR file on the server filesystem.

   - Copy WAR file to the server filesystem. CloverETL Server is packed in a WAR file of approximately 100MB size, so it cannot be uploaded directly from your local filesystem using the Admin Console.

   - go to Glassfish Admin Console

     It is accessible on URL http://localhost:4848/ by default; default username/password is "admin"/"adminadmin"

   - go to section Applications > Web Applications → Deploy button

   - Submit form

Installation of CloverETL Server License

CloverETL Server requires valid license for executing graphs. You can install CloverETL Server without license, but no graph will be executed.

Settings of configuration and license is quite similar to WebSphere configuration.

1. get file license.dat

2. set CloverETL Server parameter license.file with path to license.dat file

   - There are more ways how to do this. The most direct way is to set environment property clover_license_file. (See "configuration" section for description of all possibilities).

   - go to Glassfish Admin Console

     By default accessible on URL http://localhost:4848/ with username/password admin/adminadmin

   - go to Configuration → System Properties

     - create property named clover_license_file which value is absolute path to license.dat file on file system
• This change requires restart of Glassfish.

**Note**

CloverETL license can be **changed** any time by replacing file `license.dat`. Then you have to let CloverETL Server know, that license is changed.

• Go to **web GUI → monitoring section → license tab**
• Then click the button **reload license**.
• Or you can restart CloverETL Server application.

### JBoss

**Installation of CloverETL Server**

1. get web archive file (clover.war) with CloverETL Server application, which is built for JBoss.

2. check prerequisites

   • JDK or JRE version 1.6.x or higher
   • JBoss 6.0 or JBoss 5.1 - see [http://www.jboss.org/jbossas/downloads](http://www.jboss.org/jbossas/downloads)
   • correct memory settings for jboss java process

   Set at least 256 MB of PermGen space (512MB is recommended), and at least 1024MB for the heap memory limit. You can set these java parameters i.e. in `[jboss-home]/bin/run.conf (run.conf.bat on Windows OS):

   ```
   JAVA_OPTS="$JAVA_OPTS -XX:MaxPermSize=512m -Xms128m -Xmx1024m"
   ```

   On windows it would be analogous to the settings above.

3. configure DB data source

   Since JBoss doesn't work with embedded derby DB, there must be always some DB connection configured.

   We used MySQL in this case

   • create datasource config file `[jboss-home]/server/default/deploy/mysql-ds.xml

   ```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></password>
     </local-tx-datasource>
   </datasources>
   ```

   JNDI name must be exactly "CloverETLServerDS". The thing to do here is to set DB connection parameters (`connection-url`, `driver-class`, `user-name` and `password`) to the created database which has to be empty before the first execution. The server creates its tables itself.
Chapter 2. Installation

JNDI data source is the only way how to configure CloverETL Server DB connection in JBoss.

• put JDBC driver for your DB to the app server classpath; we copied JDBC driver mysql-connector-java-5.1.5-bin.jar to the [jboss-home]/server/default/lib

4. configure CloverETL Server

• create cloverServer.properties in some suitable directory

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

Do not change `dataSource.type` and `dataSource.jndiName` properties, but set a correct JDBC dialect according to your DB server and Chapter 16, `Configuration` (p. 81). Also set path to your license file.

5. Set system property (or environment property) `clover_config_file`.

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

The simplest way is to set java parameter i.e. in `[jboss-home]/bin/run.sh`, e.g.:

```sh
export JAVA_OPTS="-Dclover_config_file=/home/clover/config/cloverServer.properties"
```

Please do not override other settings in the `JAVA_OPTS` property. i.e memory settings as described above.

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

```bat
set "JAVA_OPTS=%JAVA_OPTS% -Dclover_config_file=C:\JBoss6\cloverServer.properties"
```

6. deploy WAR file

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

7. start jboss by `[jboss-home]/bin/run.sh` (or `run.bat` on Windows OS)

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

8. Check JBoss response and CloverETL Server response

• JBoss administration console is accessible on URL [http://localhost:8080/](http://localhost:8080/) by default. Default username/password is "admin"/"admin"

• CloverETL Server is accessible on URL [http://localhost:8080/clover](http://localhost:8080/clover) by default.

9. If you like, you can move default and example sandboxes (created automatically in the `temp` directory) to a more suitable directory on your filesystem.

• These sandboxes are created automatically during the first deployment and are located in the web-app directory, which is related to the specific deployment. If you redeployed the web application for a reason,
the directory would be recreated. That is why it is better to move the sandboxes to a location which will not change.

### Installation of CloverETL Server License

CloverETL Server requires valid license for executing graphs. You can install CloverETL Server without license, but no graph will be executed.

1. **get file license.dat**
   - If you have only clover_license.war, extract it as common zip archive and you will find license.dat file in WEB-INF subdirectory

2. **set CloverETL Server parameter license.file with path to license.dat file**
   - The best way how to configure license, is to set config property license.file in the cloverServer.properties file as described in the beginning of this section.
   - There are more ways how to do this. (See Chapter 16, Configuration (p. 81) for description of all possibilities).

3. **Change of configuration requires restart of app-server.**

   **Note**
   - CloverETL license can be **changed** any time by replacing file license.dat. Then you have to let CloverETL Server know, that license is changed.
   - Go to web GUI → monitoring section → license tab
   - Then click the button reload license.
   - Or you can restart CloverETL Server application.

### Possible issues during installation

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

#### 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.

On Linux:

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

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

As a final step, restart the application server.

On Windows OS:

Set JAVA_HOME to your JDK installation directory, e.g. C:\Program Files\java\jdk1.6.0. Optionally, set also JRE_HOME to the JRE base directory, e.g. C:\Program Files\java\jre6.
Chapter 2. Installation

Timeouts waiting for JVM

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

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

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

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

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

**clover.war as default context on Websphere (Windows OS)**

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

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

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

![Enterprise Applications](image_url)

*Figure 2.1. Clover Server as the only running application on IBM Websphere*
Chapter 2. Installation

**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.

**System variables and more than one CloverETL Server instances running on the single machine (Windows OS)**

If you are setting system variables like `clover_license_file` or `clover_config_file` on Windows OS, remember you should not be running more than one CloverETL Server. Therefore if you ever needed to run more instances at once, use other ways of setting parameters (see Chapter 16, Configuration (p. 81) for description of all possibilities) The reason is the environment variable is shared by all applications in use causing them to share configurations and fail unexpectedly.

**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 all 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, why not install your application server in a safe destination like:

```
C:\JBoss6\license.file=C:/CoverETL/Server/license.dat
```

**JAXB and early versions of JVM 1.6**

CloverETL Server contains jaxb 2.1 libraries since version 1.3. This may cause conflicts on early versions of JVM 1.6 which contain jaxb 2.0. However JDK6 Update 4 release finally contains jaxb 2.1, thus update to this or newer version of JVM solves possible conflicts.

**File system permissions**

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

**JMS API and JMS third-party libraries**

Missing JMS libraries do not cause fail of server startup, but it is issue of deployment on application server, thus it still suits to this chapter.
clover.war itself does not contain jms.jar, thus it has to be on application server's classpath. Most of the application servers have jms.jar by default, but i.e. tomcat does not. so if the JMS features are needed, the jms.jar has to be added explicitly.

If "JMS Task" feature is used, there must be third-party libraries on server's classpath as well. The same approach is recommended for JMS Reader/Writer components, even if these components allow to specify external libraries. It is due to common memory leak in these libraries which causes "OutOfMemoryError: PermGen space". 
Chapter 3. Graphs on Server Side - Sandboxes

Sandbox is a base storage unit for project. Sandbox is actually a server-side analogy to a CloverETL Designer project. Since CloverETL Designer has a connector to CloverETL Server, a designer project and a server sandbox may be linked together. This remote CloverETL Designer project looks and works like common local project, but all files are stored on the server side and all operations are performed on server side. See CloverETL Designer manual for details on configuring a connection to the server.

Technically speaking, a sandbox is a dedicated directory on the server file system. A sandbox cannot contain another sandbox. It is recommended to have one directory as sandboxes container and create a subdirectory for each sandbox. Files and directories in sandboxes are read by JVM of Application Server. Thus, all these directories must be accessible to the OS user who executes JVM of Application Server. i.e. If Apache Tomcat is executed as an OS service by "tomcat" user, all sandboxes must be accessible to this user.

In cluster mode, there are three sandbox types: "shared", "local" and "partitioned". See Chapter 22, Clustering (p. 101) for details.

![Figure 3.1. Sandboxes Section in CloverETL Server Web GUI](image)

Each sandbox is defined by following attributes:
Table 3.1. Sandbox attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Unique &quot;name&quot; of the sandbox. It is used in server APIs to identify sandbox. It must meet common rules for identifiers. It is specified by user during sandbox creation and it can be modified later. Note: modifying is not recommended, because it may be already used by some CS APIs clients.</td>
</tr>
<tr>
<td>Name</td>
<td>Sandbox name used just for display. It is specified by user during sandbox creation and it can be modified later.</td>
</tr>
<tr>
<td>Root path</td>
<td>Absolute server side file system path to sandbox root. It is specified by user during sandbox creation and it can be modified later. This attribute is used only in standalone mode. See Chapter 22, Clustering (p. 101) for details about cluster mode.</td>
</tr>
<tr>
<td>Owner</td>
<td>It is set automatically during sandbox creation. It may be modified later.</td>
</tr>
</tbody>
</table>

Figure 3.2. Sandbox Detail in CloverETL Server Web GUI

Referencing files from the graph

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 graph file.

- SANDBOX_* placeholders

  It is possible to use placeholders for paths to another sandboxes. Placeholder is constructed from sandbox ID with "SANDBOX_*" prefix. I.e. placeholder for default sandbox is: SANDBOX_default and you can use it in graph XML like this: ${SANDBOX_default}. Placeholder is replaced by path to the sandbox's root path during graph preprocessing. These absolute local filesystem paths won't work in cluster environment! It is recommended to use sandbox URL instead.

- 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. 105) for details about the sandbox URLs.
Sandbox Security and Permissions

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

Figure 3.3. Sandbox Permissions in CloverETL Server Web GUI

Permissions to 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 3.2. Sandbox permissions

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R - read</td>
<td>Users can see this sandbox in their sandboxes list.</td>
</tr>
<tr>
<td>W - write</td>
<td>Users can modify files in the sandbox through CS APIs.</td>
</tr>
<tr>
<td>X - execution</td>
<td>Users can execute graphs in this sandbox. Note: graph executed by &quot;graph event listener&quot; is actually executed by the same user as graph which is source of event. See details in &quot;graph event listener&quot;. Graph executed by schedule trigger is actually executed by the schedule owner. See details in Chapter 5, Scheduling (p. 31).</td>
</tr>
</tbody>
</table>

Sandbox Content

Sandbox should contain graphs, metadata, external connection and all related files. Files especially graph files are identified by relative path from sandbox root. Thus you need two values to identify specific graph: sandbox and path in sandbox.
Although web GUI section **sandboxes** isn't file-manager, it offers some useful features for sandbox management.

**Download sandbox in ZIP**

Select sandbox in left panel, then web GUI displays button "Download sandbox in 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.
Upload ZIP to sandbox

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

Figure 3.6. Web GUI - upload ZIP to sandbox

Figure 3.7. Web GUI - upload ZIP results
Chapter 3. Graphs on
Server Side - Sandboxes

Table 3.3. ZIP upload parameters

<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding of packed file names</td>
<td>File names which contain special characters (non ASCII) are encoded. By this select box, you choose right encoding, so filenames are decoded properly.</td>
</tr>
<tr>
<td>Overwrite existing files</td>
<td>If this switch is checked, existing file is overwritten by new one, if both of them are stored in the same path in the sandbox and both of them have the same name.</td>
</tr>
<tr>
<td>Replace sandbox content</td>
<td>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 &quot;May delete files, which are missing in uploaded ZIP&quot; to enable it.</td>
</tr>
</tbody>
</table>

Download file in ZIP

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

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

Figure 3.8. Web GUI - download file in 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

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.

Graph config properties

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

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

If neither sandbox or graph have config properties specified, defaults from main server configuration are applied.
(See Chapter 16, Configuration (p. 81) for details)

In addition, it is possible to specify additional graph parameters, which can be used as placeholders in graph XML. Please keep in mind, that these placeholders are resolved during loading and parsing of XML file, thus such graph couldn't be pooled.
### Table 3.4. Graph config parameters

<table>
<thead>
<tr>
<th>Property name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tracking_interval</td>
<td>2000</td>
<td>Interval in ms for sampling nodes status in running graph.</td>
</tr>
<tr>
<td>max_running_concurrently</td>
<td>unlimited</td>
<td>Max number of concurrently running instances of this graph.</td>
</tr>
<tr>
<td>enqueue_executions</td>
<td>false</td>
<td>Boolean value. If it is true, executions above max_running_concurrently are enqueued, if it is false executions above max_running_concurrently fail.</td>
</tr>
<tr>
<td>log_level</td>
<td>INFO</td>
<td>Log4j log level for this graph executions. (ALL</td>
</tr>
<tr>
<td>max_graph_instance_age</td>
<td>0</td>
<td>Time interval in ms which specifies how long may graph instance last in server's cache. 0 means that graph is initialized and released for each execution. Graph cannot be stored in the pool and reused in some cases (graph uses placeholders using dynamically specified parameters)</td>
</tr>
<tr>
<td>classpath</td>
<td></td>
<td>List of paths or jar files which contain external classes used in the graph (transformations, generators, JMS processors). Separator is specified by Engine property &quot;DEFAULT_PATH_SEPARATOR_REGEX&quot;. Path must always end with slash character &quot;/&quot;. Server automatically adds &quot;trans&quot; subdirectory of graphs's sandbox.</td>
</tr>
<tr>
<td>skip_check_config</td>
<td>default value is taken from engine property</td>
<td>Switch which specifies whether check config must be performed before graph execution.</td>
</tr>
<tr>
<td>password</td>
<td></td>
<td>Password for decoding of encoded DB connection passwords.</td>
</tr>
<tr>
<td>verbose_mode</td>
<td>true</td>
<td>If true, more descriptive logs of graph runs are generated.</td>
</tr>
<tr>
<td>use_jmx</td>
<td>true</td>
<td>If true, graph executor registers jmx mBean of running graph.</td>
</tr>
<tr>
<td>debug_mode</td>
<td>false</td>
<td>If true, edges with enabled debug store data into files in debug directory. See property &quot;graph.debug_path&quot;</td>
</tr>
</tbody>
</table>
Figure 3.9. Graph config properties
Chapter 4. Users and Groups

CloverETL Server implements security module, which manages users and groups. Security module may be globally switched off (see Chapter 16, Configuration (p. 81) for details), but by default it is on, and all interfaces require client authentication by username and password. Relation between users and groups is N:M, thus one user may be assigned in more groups and one group may be assigned in more users.

All relations between users and groups are configurable in web GUI in sections Users and Groups.

Both sections are accessible only for users which have "List users" ("List groups" resp.) permission. To modify users/groups "create", "edit" and "delete" permissions are necessary.

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 GUI).

However authorization (access levels to sandboxes content and privileges for operations) is still handled by Clover security module. Each user, event though logged-in using LDAP authentication, must have his own "user" record (with related groups) in CloverETL security module. So there must be the user with the same username and domain set to "LDAP". If no such user record exists, it's automatically created according to CloverETL configuration.

What does the CloverETL do to authenticate a LDAP user?

1. User specifies the LDAP credentials i.e. in login form to the web GUI
2. CloverETL Server connects to the LDAP and checks whether the user exists (it uses specified search to lookup in LDAP)
3. If the user exists in LDAP, CloverETL Server performs authentication
4. If succeeded, CloverETL Server searches for LDAP user's groups.
5. CloverETL Server checks whether the user is assigned in LDAP groups which are allowed to login to Clover.
6. Clover user record is created/updated according to current LDAP values.
7. Clover user is assigned to the Clover groups according to his current assignation to the LDAP groups.
8. User is logged-in

Configuration

By default CloverETL Server allows only its own internal mechanism for authentication. To enable authentication with LDAP, set config property "security.authentication.allowed_domains" properly. It's list of user domains which are used for authentication.

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

```java
# Implementation of context factory
security.ldap.ctx_factory=com.sun.jndi.ldap.LdapCtxFactory
# timeout for all queries sent to LDAP server
security.ldap.timeout=5000
# limit for number of records returned from LDAP
security.ldap.records_limit=50
# URL of LDAP server
security.ldap.url=ldap://hostname:port
# Some generic UserDN which allows lookup for the user and groups.
security.ldap.userDN=
# Password for the user specified above
security.ldap.password=
```

Configuration of user lookup

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 [2] in the login process above)

```java
# 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.
# Please 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
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)

```java
security.ldap.user_search.attribute.firstname=fn
security.ldap.user_search.attribute.lastname=sn
security.ldap.user_search.attribute.email=mail
```
Chapter 4. Users and Groups

Clover user record will be assigned to the clover groups according to the LDAP groups found by following search. (Groups check is performed during each login) So the following properties define search for the groups which the user is member of. (step [4] in the login process above)

```
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
# 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
```

It's also possible to specify LDAP groups which are able to login to Clover. (step [5] in the login process above)

```
# Semicolon separated list of LDAP group DNs (distinguished names).
# LDAP user must be assigned to one or more of these groups, otherwise new clover user can't be created.
# Special value "_ANY_" disables this check and basically any LDAP user may login.
# If the LDAP group DNs are configured, also security.ldap.groups_search.* properties must be configured.
# value could be e.g. "cn=developers,dc=company,dc=com;cn=admins,dc=company,dc=com"
security.ldap.allowed_ldap_groups=_ANY_
```

**Web GUI section Users**

This section is intended to users management. It offers features in dependence of 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
- delete user
- assign user to groups - Assignment to groups gives user proper permissions

*Table 4.1. After default installation above empty DB, there are two users created*

<table>
<thead>
<tr>
<th>User name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clover</td>
<td>Clover user has admin permissions, thus default password “clover” should be changed after installation.</td>
</tr>
<tr>
<td>system</td>
<td>System user is used by application instead of common user, when no other user can be used. i.e. when security is globally switched off. This user cannot be removed and it is impossible to login as this user.</td>
</tr>
</tbody>
</table>
Chapter 4. Users and Groups

Figure 4.1. Web GUI - section "Users"

Table 4.2. User attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Common user identifier. Must be unique, cannot contain spaces or special characters, just letters and numbers.</td>
</tr>
<tr>
<td>password</td>
<td>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.</td>
</tr>
<tr>
<td>first name</td>
<td></td>
</tr>
<tr>
<td>last name</td>
<td></td>
</tr>
<tr>
<td>email</td>
<td>Email which may be used by CloverETL administrator or by CloverETL server for automatic notifications. See Task - Send Email (p. 41) for details.</td>
</tr>
</tbody>
</table>

Edit user record

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

Figure 4.2. 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.
Chapter 4. Users and Groups

Figure 4.3. Web GUI - change password

Group assignment

Assignment to groups gives user proper permissions. Only logged user with permission “Groups assignment” can access this form and specify groups which the user is assigned in. See Web GUI section Groups (p. 28) for details about permissions.

Figure 4.4. Web GUI - groups assignment

Web GUI section Groups

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

There are independent levels of permissions implemented in CloverETL Server

- permissions to Read/Write/eXecute in sandboxes - sandbox owner can specify different permissions for different groups. See Sandbox Security and Permissions (p. 17) 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 Chapter 15, Launch Service (p. 74) for details.
Table 4.3. Default groups created during installation

<table>
<thead>
<tr>
<th>Group name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>admins</td>
<td>This group has operation permission &quot;all&quot; assigned, which means, that it has unlimited permission. Default user &quot;clover&quot; is assigned to this group, which makes him administrator.</td>
</tr>
<tr>
<td>all users</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

Groups permissions

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

Thus for "admin" group just "all" permission is assigned, every single permission in sub tree is assigned automatically.
Figure 4.7. Tree of permissions
Chapter 5. Scheduling

Scheduling allows user to create his own timetable for operations which he does not want to trigger manually. Each schedule represents separated timetable and basically its specification WHEN to do something and WHAT to do.

In cluster environment, scheduling is processed only on master node, thus tasks are triggered only on master node.

Figure 5.1. Web GUI - section "Scheduling" - create new Timetable Setting

This section should describe how to specify WHEN schedule should be triggered. Please keep in mind, that exact trigger times are not guaranteed. There may be couple of seconds delay. Schedule itself can be specified in different ways.

- **Onetime Schedule** (p. 31)
- **Periodical schedule by Interval** (p. 32)
- **Periodical schedule by timetable (Cron Expression)** (p. 33)

### Onetime Schedule

It is obvious, that this schedule is triggered just once.

**Table 5.1. Onetime schedule attributes**

<table>
<thead>
<tr>
<th>Type</th>
<th>&quot;onetime&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start date/time</td>
<td>Date and time, specified with minutes precision.</td>
</tr>
</tbody>
</table>
Chapter 5. Scheduling

**Figure 5.2. Web GUI - onetime schedule form**

**Figure 5.3. 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 5.2. Periodical schedule attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>&quot;periodic&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodicity</td>
<td>&quot;interval&quot;</td>
</tr>
<tr>
<td>Start date/time</td>
<td>Date and time, specified with minutes precision.</td>
</tr>
<tr>
<td>End date/time</td>
<td>Date and time, specified with minutes precision.</td>
</tr>
<tr>
<td>Interval in minutes</td>
<td>Specifies interval between two trigger times. Next task is triggered even if previous task is still running.</td>
</tr>
<tr>
<td>Fire misfired ASAP switch</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Figure 5.4. Web GUI - periodical schedule form**

**Periodical schedule by timetable (Cron Expression)**

Timetable is specified by powerful (but a little bit tricky) cron expression.

Table 5.3. Cron periodical schedule attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>&quot;periodic&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periodicity</td>
<td>&quot;interval&quot;</td>
</tr>
<tr>
<td>Start date/time</td>
<td>Date and time, specified with minutes precision.</td>
</tr>
<tr>
<td>End date/time</td>
<td>Date and time, specified with minutes precision.</td>
</tr>
<tr>
<td>Cron expression</td>
<td>Cron is powerful tool, which uses its own format for scheduling. This format is well known among UNIX administrators. i.e. &quot;0 0/2 4-23 * * ?&quot; means &quot;every 2 minutes between 4:00am and 11:59pm&quot;.</td>
</tr>
<tr>
<td>Fire misfired ASAP switch</td>
<td>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.</td>
</tr>
</tbody>
</table>
Chapter 5. Scheduling

Figure 5.5. Cron periodical schedule form

Tasks

Task basically specifies WHAT to do at trigger time. There are several tasks implemented for schedule and for graph event listener as follows:

- Task - Execution of Graph (p. 35)
- Task - Kill Graph (p. 35)
- Task - Execution of Shell Command (p. 36)
- Task - Send Email (p. 37)
- Task - Execute Groovy Code (p. 37)
- Task - Archive Records (p. 38)

We expect, that some more task implementation will be needed, i.e. task type "Execution of java code", etc.
Task - Execution of Graph

Table 5.4. Attributes of "Graph execution" task

<table>
<thead>
<tr>
<th>Task type</th>
<th>&quot;execute graph&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandbox</td>
<td>This select box contains sandboxes which are readable for logger user. Select sandbox which contains graph to execute.</td>
</tr>
<tr>
<td>Graph</td>
<td>This select box is filled by all graphs accessible in selected sandbox.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Key-value pairs which are passed to the executed graph as parameters. Besides, if this task is triggered by &quot;graph event&quot;, you can specify parameters from the graph which is &quot;graph event&quot; source. These parameters are passed to executed graph, i.e. event source graph has these parameters: paramName2 with value &quot;val2&quot;, paramName3 with value &quot;val3&quot;, paramNameX. Task &quot;graph execution&quot; has &quot;Parameters&quot; attribute set like this:</td>
</tr>
</tbody>
</table>

```
  paramName1=paramValue1
  paramName2=
  paramName3
  paramName4
```

So executed graph gets these parameters and values: paramName1 with value "paramValue1" (specified by task) paramName2 with value "" (empty string specified by task overrides event source parameters) paramName3 with value "val3" (value is taken from event source graph) These parameters aren't passed: paramName4 isn't passed, since it does not have any value in event source graph. paramNameX isn't passed, since it is not specified among the parameters to pass in the task Event parameters like "event_run_result", "event_run_id" etc. are passed to the graph without limitations.

![Figure 5.6. Web GUI - Graph execution task](image)

Task - Kill Graph

This task, when activated kills/aborts specified graph, if it is currently running.
Chapter 5. Scheduling

Table 5.5. Attributes of "Kill graph" task

<table>
<thead>
<tr>
<th>Task type</th>
<th>&quot;kill graph&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kill source of event</td>
<td>If this switch is on, task will kill graph which is source of the event, which activated this task. Attributes Sandbox and graph are ignored.</td>
</tr>
<tr>
<td>Sandbox</td>
<td>Select sandbox which contains graph to kill. This attribute takes place only when &quot;Kill source of event&quot; switch is off.</td>
</tr>
<tr>
<td>Graph</td>
<td>This select box is filled by all graphs accessible in selected sandbox. All instances of selected graph, whose are currently running will be killed. This attribute takes place only when &quot;Kill source of event&quot; switch is off.</td>
</tr>
</tbody>
</table>

![Figure 5.7. Web GUI - "Kill graph"](image)

Task - Execution of Shell Command

Table 5.6. Attributes of "Shell command" task

<table>
<thead>
<tr>
<th>Task type</th>
<th>“shell command”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command line</td>
<td>Command line for execution of external process.</td>
</tr>
<tr>
<td>Working directory</td>
<td>Working directory for process. If not set, working directory of application server process is used.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Timeout in milliseconds. After period of time specified by this number, external process is terminated and all results are logged.</td>
</tr>
</tbody>
</table>

![Figure 5.8. Web GUI - shell command](image)
Task - Send Email

This task is very useful, but for now only as response for graph events. This feature is very powerful for monitoring. (see Chapter 6, Graph Event Listeners (p. 40) for description of this task type).

Note: It seems useless to send emails periodically, but it may send current server status or daily summary. These features will be implemented in further versions.

Task - Execute Groovy Code

This type of task allows execute code written in script language Groovy. It is possible to use some variables. Only parameter of this task is source code of written in Groovy.

Table 5.7. List of variables available in Groovy code

<table>
<thead>
<tr>
<th>variable</th>
<th>class available</th>
<th>description</th>
<th>availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>com.cloveretl.server.events.AbstractServerEvent</td>
<td>every time</td>
<td></td>
</tr>
<tr>
<td>task</td>
<td>com.cloveretl.server.persistent.Task</td>
<td>every time</td>
<td></td>
</tr>
<tr>
<td>now</td>
<td>java.util.Date</td>
<td>current time</td>
<td>every time</td>
</tr>
<tr>
<td>parameters</td>
<td>java.util.Properties</td>
<td>Properties of task</td>
<td>every time</td>
</tr>
<tr>
<td>user</td>
<td>com.cloveretl.server.persistent.User</td>
<td>Same as event.getUser()</td>
<td>every time</td>
</tr>
<tr>
<td>run</td>
<td>com.cloveretl.server.persistent.RunRecord</td>
<td>When the event is instance of GraphServerEvent</td>
<td></td>
</tr>
<tr>
<td>tracking</td>
<td>com.cloveretl.server.persistent.TrackingGraph</td>
<td>When the event is instance of GraphServerEvent</td>
<td></td>
</tr>
<tr>
<td>sandbox</td>
<td>com.cloveretl.server.persistent.Sandbox</td>
<td>When the event is instance of GraphServerEvent</td>
<td></td>
</tr>
<tr>
<td>schedule</td>
<td>com.cloveretl.server.persistent.Schedule</td>
<td>When the event is instance of ScheduleServerEvent</td>
<td></td>
</tr>
<tr>
<td>servletContext</td>
<td>javax.servlet.ServletContext</td>
<td>every time</td>
<td></td>
</tr>
<tr>
<td>cloverConfiguration</td>
<td>com.cloveretl.server.spring.CloverConfiguration</td>
<td>every time</td>
<td></td>
</tr>
<tr>
<td>serverFacade</td>
<td>com.cloveretl.server.facade.api.ServerFacade</td>
<td>every time</td>
<td></td>
</tr>
<tr>
<td>sessionToken</td>
<td>String</td>
<td>Valid session token of the user who owns the event. It is useful for authorisation to the facade interface.</td>
<td>every time</td>
</tr>
</tbody>
</table>

Variables run, tracking and sandbox are available only if event is instance of GraphServerEvent class. Variable schedule is available only for ScheduleServerEvent as event variable class.

Example of use Groovy script

This example shows script which writes text file describing finished graph. It shows use of 'run' variable.
import com.cloveretl.server.persistent.RunRecord;
String dir = "~/tmp/";
RunRecord rr = (RunRecord)run;

String fileName = "report"+rr.getId()+"_finished.txt";
FileWriter fw = new FileWriter(new File(dir+fileName));
fw.write("Run ID :"+rr.getId()+"\n");
fw.write("Graph ID :"+rr.getGraphId()+"\n");
fw.write("Sandbox :"+rr.getSandbox().getName()+"\n");
fw.write("Start time :"+rr.getStartTime()+"\n");
fw.write("Stop time :"+rr.getStopTime()+"\n");
fw.write("Duration :"+rr.getDurationString()+"\n");
fw.write("Final status :"+rr.getFinalStatus()+"\n");
fw.close();

Task - Archive Records

As name suggests, this task can archive (or delete) obsolete records from DB.

Table 5.8. Attributes of "archive records" task

<table>
<thead>
<tr>
<th>Task type</th>
<th>&quot;archivator&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Older then</td>
<td>Time period (in minutes) - it specifies which records are evaluated as obsolete. Records older then the specified interval are stored in archives.</td>
</tr>
<tr>
<td>Archivator type</td>
<td>There are two possible values: &quot;archive&quot; or &quot;delete&quot;. Delete removes records without any possibility of UNDO operation. Archive removes records from DB, but creates ZIP package with CSV files containing deleted data.</td>
</tr>
<tr>
<td>Output path for archives</td>
<td>This attribute makes sense only for &quot;archive&quot; type.</td>
</tr>
<tr>
<td>Include executions history</td>
<td>If status is selected, only run records with specified status will be archived. It is useful i.e. If you want to delete records for successfully finished graphs, but you are interested in failed graphs.</td>
</tr>
<tr>
<td>Run record with status</td>
<td>If checked, archivator will include run records. Log files of graph runs are included as well.</td>
</tr>
<tr>
<td>Task types</td>
<td>If this task type is selected, only logs for selected task type are archived.</td>
</tr>
<tr>
<td>Task result mask</td>
<td>Mask applied to task log result attribute. Only records whose result meets this mask are archived. Specify string without any wildcards. Each task log which contains specified string in the &quot;result&quot; attribute will be deleted/archived. Case sensitivity depends on database collation.</td>
</tr>
<tr>
<td>Include debug files</td>
<td>If checked, archivator removes all graph debug files older then given timestamp defined in &quot;Older than&quot; attribute.</td>
</tr>
<tr>
<td>Include dictionary files</td>
<td>If checked, archivator removes all dictionary temporary files older then given timestamp defined in &quot;Older than&quot; attribute.</td>
</tr>
</tbody>
</table>
Chapter 5. Scheduling

Figure 5.9. Web GUI - archive records
Chapter 6. Graph Event Listeners

Graph event listener is powerful feature, which allows user to monitor success of failure of graph executions. It is also possible to create relations between executions, or execute backup script in dependence of graph success or failure.

In cluster environment, event exists only on cluster node, which runs graph thus task is triggered on the same node.

Graph Events

Each event carries properties of graph, which is source of event. If there is a event listener specified, task may use these properties, i.e. next graphs in the chain may use "event_file_name" placeholder which activated first graph in the chain. Graph properties, which are set specifically for each graph run (i.e. RUN_ID), are overridden by last graph.

For now, there are these types of graph events:

- **graph started** (p. 40)
- **graph finished OK** (p. 40)
- **graph error** (p. 40)
- **graph aborted** (p. 40)
- **graph timeout** (p. 40)
- **graph status unknown** (p. 41)

**graph started**

Event of this type is created, when graph is successfully executed. It means, that threads of graph nodes and watchdog are running.

**graph finished OK**

Event of this type is created, when all phases and nodes of graph are finished with status FINISHED_OK.

**graph error**

Event of this type is created, when graph cannot be executed from any reason, or when any node of graph fails.

**graph aborted**

Since 1.2.1

Event of this type is created, when graph is explicitly aborted.

**graph timeout**

Event of this type is created, when graph runs longer then specified interval. Thus you have to specify "Graph timeout interval" attribute for each listener of graph timeout event. You can specify this interval in seconds or in minutes or in hours.
Chapter 6. Graph Event Listeners

Since 1.3.

Event of this type is created, when server starts and there is graph with undefined status in the executions history. Undefined status means, that server has been killed during graph run. Server automatically changes state of graph to "Not Available" and sends 'graph status unknown' event. Please note, that this works just for executions, which have persistent record in executions history. It is possible to execute transformation without persistent record in executions history, typically for better performance of fast running transformations (i.e. using Launch Services).

Listener

User may create listener for specified event type and graph (or all graphs in sandbox). Listener is actually connection between graph event and task, where graph event specifies WHEN and task specifies WHAT to do.

So progress is like this:

• event is created
• listeners for this event are notified
• each listener performs related task

Tasks

Task types "execute shell command", "execute graph" and "archivator" are described in section "scheduling" see this section for details about these task types. There is one more task type, which is useful especially with graph event listeners, thus it is described here. It is task type "send email".

Note: You can use task of any type for both scheduling and graph event listener. Description of task types is divided into two sections just to show the most obvious use cases.

• Task - Send Email (p. 41)
• Task - JMS Message (p. 44)

Task - Send Email

This type of task is useful for notifications about result of graph execution. i.e. you can create listener with this task type to be notified about each failure in specified sandbox or failure of particular graph.
Table 6.1. Attributes of "Send email" task

<table>
<thead>
<tr>
<th>Task type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task type</td>
<td>&quot;email&quot;</td>
</tr>
<tr>
<td>Email pattern</td>
<td>This select box contains all predefined email patterns. If user chooses any of them, all fields below are automatically filled by values from pattern.</td>
</tr>
<tr>
<td>To</td>
<td>Recipient's email address. It is possible to specify more addresses separated by comma. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>Cc</td>
<td>Cc stands for 'carbon copy'. Copy of email will be delivered to these addresses. It is possible to specify more addresses separated by comma. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>BCc</td>
<td>Bcc: stands for 'Blind carbon copy'. It is the same as Cc, but the others recipients aren't aware, that these recipients get copy of email.</td>
</tr>
<tr>
<td>Reply-to (Sender)</td>
<td>Email address of sender. It must be valid address according to SMTP server. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>Subject</td>
<td>Email subject. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>Plain text</td>
<td>Body of email in plain text. Email is created as multipart, so HTML body should have a precedence. Plain text body is only for email clients which do not display HTML. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>HTML</td>
<td>Body of email in HTML. Email is created as multipart, so HTML body should have a precedence. Plain text body is only for email clients which do not display HTML. It is also possible to use placeholders. See Placeholders (p. 43) for details.</td>
</tr>
<tr>
<td>Log file as attachment</td>
<td>If this switch is checked, email will have an attachment with packed log file of related graph execution.</td>
</tr>
</tbody>
</table>
Figure 6.2. Web GUI - send email

Note: Do not forget to configure connection to SMTP server (See Chapter 16, Configuration (p. 81) for details).

**Placeholders**

Place holder may be used in some fields of tasks. They are especially useful for email tasks, where you can generate content of email according to context variables.

*Note: In most cases, you can avoid this by using email patterns (See Email task for details)*

These fields are preprocessed by Apache Velocity templating engine. See Velocity project URL for syntax description [http://velocity.apache.org/](http://velocity.apache.org/)

There are several context variables, which you can use in place holders and even for creating loops and conditions.

* **event**
* **now**
* **user**
* **run**
* **sandbox**

Some of them may be empty in dependence of occasion which field is processed in. I.e. If task is processed because of graph event, then **run** and **sandbox** variables contain related data, otherwise they are empty,
Table 6.2. Placeholders useful in email templates

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>now</td>
<td>Current date-time</td>
</tr>
<tr>
<td>user</td>
<td>User, who caused this event. It may be owner of schedule, or someone who executed graph. Contains sub-properties, which are accessible using dot notation (i.e. <code>${user.email}</code>) email, username, firstName, lastName, groups (list of values)</td>
</tr>
<tr>
<td>run</td>
<td>Data structure describing one single graph execution. Contains sub-properties, which are accessible using dot notation (i.e. <code>${run.graphId}</code>)  graphId, finalStatus, startTime, stopTime, errNode, errMessage, errException, logLocation</td>
</tr>
<tr>
<td>tracking</td>
<td>Data structure describing status of components in graph execution. Contains sub-properties, which are accessible using Velocity syntax for loops and conditions.</td>
</tr>
<tr>
<td></td>
<td>#if (${tracking})</td>
</tr>
<tr>
<td></td>
<td>&lt;table border=&quot;1&quot; cellpadding=&quot;2&quot; cellspacing=&quot;0&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>#foreach ($phase in $tracking.trackingPhases)</td>
</tr>
<tr>
<td></td>
<td>&lt;tr&gt;&lt;td&gt;phase: ${phase.phaseNumber}&lt;/td&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;td&gt;${phase.executeTime} ms&lt;/td&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;td&gt;${phase.executeTime} ms&lt;/td&gt;</td>
</tr>
</tbody>
</table>
|               | </tr>
|               | #foreach ($node in $phase.trackingNodes) |
|               | <tr><td>${node.nodeName}</td> |
|               | <td>${node.result}</td> |
|               | </tr>
|               | #foreach ($port in $node.trackingPorts) |
|               | <tr><td>${port.portType}:${port.index}</td> |
|               | <td>${port.totalBytes} B</td> |
|               | <td>${port.totalRows} rows</td></tr>
|               | #end
|               | #end
|               | #end
|               | </table> |
|               | #end |
| sandbox       | Data structure describing sandbox containing executed graph. Contains sub-properties, which are accessible using dot notation (i.e. `${sandbox.name}`) name, code, rootPath |
| schedule      | Data structure describing schedule which triggered this task. Contains sub-properties, which are accessible using dot notation (i.e. `${schedule.description}`) description, startTime, endTime, lastEvent, nextEvent, fireMisfired |

**Task - JMS Message**

This type of task is useful for notifications about result of graph execution. I.e. you can create graph event listener with this task type to be notified about each failure in specified sandbox or failure of particular graph.

JMS messaging requires JMS API (jms.jar) and third-party libraries. All these libraries must be available on application server classpath. Some application servers contain these libraries by default, some do not, thus the libraries must be added explicitly.
Table 6.3. Attributes of JMS message task

<table>
<thead>
<tr>
<th>Task type</th>
<th>&quot;JMS message&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial context class name</td>
<td>Full class name of javax.naming.InitialContext implementation. Each JMS provider has own implementation. i.e. for Apache MQ it is &quot;org.apache.activemq.jndi.ActiveMQInitialContextFactory&quot;. If it is empty, server uses default initial context</td>
</tr>
<tr>
<td>Connection factory JNDI name</td>
<td>JNDI name of connection factory. Depends on JMS provider.</td>
</tr>
<tr>
<td>Destination</td>
<td>JNDI name of message queue/topic on the server</td>
</tr>
<tr>
<td>Username</td>
<td>Username for connection to JMS message broker</td>
</tr>
<tr>
<td>Password</td>
<td>Password for connection to JMS message broker</td>
</tr>
<tr>
<td>URL</td>
<td>URL of JMS message broker</td>
</tr>
<tr>
<td>JMS pattern</td>
<td>This select box contains all predefined JMS message patterns. If user chooses any of them, text field below is automatically filled by value from pattern.</td>
</tr>
<tr>
<td>Text</td>
<td>Body of JMS message. It is also possible to use placeholders. See [Placeholders](p. 43) for details.</td>
</tr>
</tbody>
</table>

![Edit event browser](image)

Figure 6.3. Web GUI - Task JMS message editor

Use cases

Possible use cases are the following:

- **Execute graphs in chain** (p. 46)
- **Email notification about graph failure** (p. 46)
- **Email notification about graph success** (p. 47)
- **Backup of data processed by graph** (p. 47)
Execute graphs in chain

Let’s say, that we have to execute graph B, only if another graph A finished without any error. So there is some kind of relation between these graphs. We can achieve this behaviour by creating graph event listener. We create listener for event graph finished OK of graph A and choose task type execute graph with graph B specified for execution. And that is it. If we create another listener for graph B with task execute graph with graph C specified, it will work as chain of graphs.

Figure 6.4. Event source graph isn’t specified, thus listener works for all graphs in specified sandbox

Email notification about graph failure

Figure 6.5. Web GUI - email notification about graph failure
Email notification about graph success

Figure 6.6. Web GUI - email notification about graph success

Backup of data processed by graph

Figure 6.7. Web GUI - backup of data processed by graph
Chapter 7. JMS messages listeners

Since 2.10

This feature allows you to specify listener for incoming JMS messages. Such listener can then process one of predefined tasks as usual for all event listeners. So for each listener user specifies source of JMS messages (JMS Topic or JMS Queue) and task which will be processed as a result of each incoming JMS message.

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.

JMS itself is quite complex topic beyond of scope of this document. Detail information about it can be found on Sun web site: http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JMS6.html
### Table 7.1. Attributes of JMS message task

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID to handle the event</td>
<td>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.</td>
</tr>
<tr>
<td>Initial context class name</td>
<td>Full class name of javax.naming.InitialContext implementation. Each JMS provider has own implementation. i.e. for Apache MQ it is &quot;org.apache.activemq.jndi.ActiveMQInitialContextFactory&quot;. If it is empty, server uses default initial context. Specified class must be on web-app classpath or application-server classpath. It is usually included in one library with JMS API implementation for each specific JMS broker provider.</td>
</tr>
<tr>
<td>Connection factory JNDI name</td>
<td>JNDI name of connection factory. Depends on JMS provider.</td>
</tr>
<tr>
<td>Destination JNDI name</td>
<td>JNDI name of message queue/topic on the server</td>
</tr>
<tr>
<td>Username</td>
<td>Username for connection to JMS message broker</td>
</tr>
<tr>
<td>Password</td>
<td>Password for connection to JMS message broker</td>
</tr>
<tr>
<td>URL</td>
<td>URL of JMS message broker</td>
</tr>
<tr>
<td>Durable subscriber (only for Topics)</td>
<td>If it is false, message consumer is connected to the broker as &quot;non-durable&quot;, so it receives only messages which are sent while the connection is active. Other messages are lost. If it is true, consumer is subscribed as &quot;durable&quot; 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 &quot;JMS message listener&quot; 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 &quot;ClientId&quot; specified. This attribute can be set in different ways in dependence of JMS provider. I.e. for ActiveMQ, it is set as URL parameter tcp://localhost:1244?jms.clientID=TestClientID</td>
</tr>
<tr>
<td>Message selector</td>
<td>This &quot;query string&quot; can be used as specification of conditions for filtering incoming messages. Syntax is well described on Java EE API web site: <a href="http://java.sun.com/j2ee/1.4/docs/api/javax/jms/Message.html">http://java.sun.com/j2ee/1.4/docs/api/javax/jms/Message.html</a> 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.</td>
</tr>
<tr>
<td>Groovy code</td>
<td>Groovy code may be used for additional message processing and/or for refusing message. Both features are described below.</td>
</tr>
</tbody>
</table>

### Optional Groovy code

Groovy code may be used for additional message processing or for refusing message.

- **Additional message processing** Groovy code may modify/add/remove values stored in containers "properties" and "data".

- **Refuse/acknowledge the message** if Groovy code returns Boolean.FALSE, message is refused. Otherwise, message is acknowledged. Refused message may be redelivered, however JMS broker should have configured some limit for redelivering messages. If groovy code throws an exception, it is considered as coding error and
JMS message is NOT refused because of it. So if the message refusal is directed by some exception, it must be handled in groovy.

**Table 7.2. Variables accessible in groovy code**

<table>
<thead>
<tr>
<th>type</th>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.jms.Message</td>
<td>msg</td>
<td>instance of JMS message</td>
</tr>
<tr>
<td>java.util.Properties</td>
<td>properties</td>
<td>See below for details. Contains values (String or converted to String) read from message and it is passed to the task which may use them somehow. I.e. task &quot;execute graph&quot; passes these parameters to the executed graph.</td>
</tr>
<tr>
<td>java.util.Map&lt;String, Object&gt;</td>
<td>data</td>
<td>See below for details. Contains values (Object, Stream, ..) read or proxied from the message instance and it is passed to the task which may use them somehow. I.e. task &quot;execute graph&quot; passes it to the executed graph as &quot;dictionary entries&quot;.</td>
</tr>
<tr>
<td>javax.servlet.ServletContext</td>
<td>servletContext</td>
<td>instance of ServletContext</td>
</tr>
<tr>
<td>javax.jms.Message</td>
<td>msg</td>
<td>instance of JMS message</td>
</tr>
<tr>
<td>String</td>
<td>sessionToken</td>
<td>sessionToken, needed for calling serverFacade methods</td>
</tr>
</tbody>
</table>

**Message data available for further processing**

JMS message is processed and data it contains is stored basically in two data structures. "properties" and "data"
### Table 7.3. "properties" elements

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jms_prop_[property key]</td>
<td>For each message property is created one entry, where &quot;key&quot; is made of prefix &quot;jms_prop_&quot; and property key.</td>
</tr>
<tr>
<td>jms_map_[map entry key]</td>
<td>If the message is instance of MapMessage, for each map entry is created one entry, where &quot;key&quot; is made of prefix &quot;jms_map_&quot; and map entry key. Values are converted to String.</td>
</tr>
<tr>
<td>jms_text</td>
<td>If the message is instance of TextMessage, this property contains content of the message.</td>
</tr>
<tr>
<td>jms_msg_class</td>
<td>Class name of message implementation</td>
</tr>
<tr>
<td>jms_msg_correlationId</td>
<td>Correlation ID is either provider-specific message ID or application-specific String value</td>
</tr>
<tr>
<td>jms_msg_destination</td>
<td>The JMSDestination header field contains the destination to which the message is being sent.</td>
</tr>
<tr>
<td>jms_msg_messageId</td>
<td>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.</td>
</tr>
<tr>
<td>jms_msg_replyTo</td>
<td>Destination to which a reply to this message should be sent.</td>
</tr>
<tr>
<td>jms_msg_type</td>
<td>Message type identifier supplied by the client when the message was sent.</td>
</tr>
<tr>
<td>jms_msg_deliveryMode</td>
<td>The DeliveryMode value specified for this message.</td>
</tr>
<tr>
<td>jms_msg_expiration</td>
<td>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.</td>
</tr>
<tr>
<td>jms_msg_priority</td>
<td>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.</td>
</tr>
<tr>
<td>jms_msg_redelivered</td>
<td>&quot;true&quot; if this message is being redelivered.</td>
</tr>
<tr>
<td>jms_msg_timestamp</td>
<td>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.</td>
</tr>
</tbody>
</table>

Please note, that all values in "properties" structure are of String type, nevertheless it is number or text.

### Table 7.4. "data" elements

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>Instance of javax.jms.Message</td>
</tr>
<tr>
<td>jms_data_stream</td>
<td>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.</td>
</tr>
<tr>
<td>jms_data_text</td>
<td>Instance of String. Only for TextMessage and ObjectMessage, where payload object is instance of String.</td>
</tr>
</tbody>
</table>

"data" container is passed to the task which may use them somehow according to its implementation. I.e. task "execute graph" passes it to the executed graph as "dictionary entries". Please note that it is not serializable, thus if the task is relying on it, it can be processed properly only on the same cluster node.

Dictionary entries can be used in some of graph component attributes. I.e. in fileURL attribute like this: "dict:jms_data_stream:discrete". So the reader reads data directly from incoming JMS message using this proxy stream.
Chapter 8. Universal event listeners

Since 2.10

This feature allows you to specify Groovy code, which decides when the event is created. Subsequently specified task is processed. So for each listener user specifies Groovy source code and task which will be processed if groovy code decides to.

Table 8.1. Attributes of Universal message task

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID to handle the event</td>
<td>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.</td>
</tr>
<tr>
<td>Interval of check in seconds</td>
<td>Periodicity of Groovy code execution.</td>
</tr>
<tr>
<td>Groovy code</td>
<td>Groovy code is used for deciding whether the event should be created or not. See below for details about groovy code.</td>
</tr>
</tbody>
</table>

Groovy code

Groovy code is used for deciding whether the event should be created or not.

i.e. it may do some checks of data sources, which are vital for execution of graph. Or it may do some complex checks of running graph and make decision to kill it. It may call CloverETL Server core functions using ServerFacade interface, which is described in its own chapter.

Creating "event" is simple. If Groovy code returns Boolean.TRUE, event is created and specified task is processed. Otherwise, nothing happens. If groovy code throws an exception, it is considered as coding error and event is NOT created because of it. So if it is necessary, the exceptions must be handled in groovy code.

Table 8.2. Variables accessible in groovy code

<table>
<thead>
<tr>
<th>type</th>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.util.Properties</td>
<td>properties</td>
<td>Empty container which may be filled by String-String key-value pairs in your Groovy code. It is passed to the task which may use them somehow. I.e. task &quot;execute graph&quot; passes these parameters to the executed graph.</td>
</tr>
<tr>
<td>java.util.Map&lt;String, Object&gt;</td>
<td>data</td>
<td>Empty container which may be filled by 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 &quot;execute graph&quot; passes it to the executed graph as &quot;dictionary entries&quot;. Please note that it is not serializable, thus if the task is relying on it, it can be processed properly only on the same cluster node.</td>
</tr>
<tr>
<td>javax.servlet.ServletContext</td>
<td>servletContext</td>
<td>instance of ServletContext</td>
</tr>
<tr>
<td>String</td>
<td>sessionToken</td>
<td>sessionToken, needed for calling serverFacade methods</td>
</tr>
</tbody>
</table>
Chapter 9. Manual task execution

Since 3.1

Manual task execution allows user to invoke task processing. Task is entity which describes how to react to some source event. So normally task is processed only as a response to some source event. Since 3.1 user can manually invoke task processing.

In addition user can specify some parameters to simulate source event which would normally trigger task processing. Following figure displays how could be simulated "file event". Parameters for various event sources are listed in section "Graph parameters"

Figure 9.1. Web GUI - "Manual task execution" section
Chapter 10. File event listeners

Since 1.3

File event listener allows system to monitor changes on server filesystem. User may define, which filesystem resource should be observed as a source of file event. User also specifies task, which should be processed as reaction to change on filesystem.

There is process which performs checks for changes on file system. This process works with preconfigured periodicity, thus there is minimal interval which for checks. You can set this minimal interval by clover property "clover.event.fileCheckMinInterval".

In cluster environment, each event listener has attribute "node ID" which specifies cluster node, which checks its local filesystem. In "standalone" environment, "node ID" attribute is ignored.

Observed file

Observed file is specified by directory path and file name pattern.

User may specify just one exact file name or file name pattern for observing more matching files in specified directory. If there are more changed files matching the pattern, separated event is triggered for each of these files.

There are three ways how to specify file name pattern of observed file(s)

- **Exact match** (p. 54)
- **Wildcards** (p. 54)
- **Regular expression** (p. 55)

**Exact match**

You specify exact name of the observed file.

**Wildcards**

You can use wildcards common in most operating systems (*, ?, etc.)

---

Figure 10.1. Web GUI - "File event listeners" section
• * - Matches zero or more instances of any character
• ? - Matches one instance of any character
• [...] - Matches any of characters enclosed by the brackets
• \ - Escape character

Examples
• *.csv - Matches all CSV files
• input_*.csv - Matches i.e. input_001.csv, input_9.csv
• input_???.csv - Matches i.e. input_001.csv, but does not match input_9.csv

Regular expression

Examples
• (.*?\.(jpg|jpeg|png|gif)$ - Matches image files

Notes
• It is strongly recommended to use absolute paths. It is possible to use relative path, but working directory depends on application server.
• Use forward slashes as file separators, even on MS Windows OS. Backslashes might be evaluated as escape sequences.

File Events

For each listener you have to specify event type, which you are interested in.

There are four types of file events:
• file APPEARANCE (p. 55)
• file DISAPPEARANCE (p. 55)
• file SIZE (p. 56)
• file CHANGE_TIME (p. 56)

file APPEARANCE

Event of this type occurs, when the observed file is created or copied from another location between two checks. Please keep in mind, that event of this type occurs immediately when new file is detected, regardless it is complete or not. Thus task which may need complete file is executed when file is still incomplete. Recommended approach is to save file to the different location and when it is complete, move/rename to observed location where CloverETL Server may detect it. File moving/renameing should be atomic operation.

Event of this type does not occur when the file has been updated (change of timestamp or size) between two checks. Appearance means that the file didn't exist during previous check and it exists now, during current check.

file DISAPPEARANCE

Event of this type occurs, when observed file is deleted or moved to another location between two checks.
Chapter 10. File event listeners

**file SIZE**

Event of this type occurs when the size of the observed file has changed between two checks. Event of this type is never produced when file is created or removed. File must exist during both checks.

**file CHANGE_TIME**

Event of this type occurs, when change time of observed file has changed between two checks. Event of this type is never produced when file is created or removed. File must exist during both checks.

### Check interval, Task and Use cases

- User may specify minimal time interval between two checks. It is specified in seconds.
- Each listener defines task, which will be processed as the reaction for file event. All task types and theirs attributes are described in section Scheduling and GraphEventListeners
  - Graph Execution, when file with input data is accessible
  - Graph Execution, when file with input data is updated
  - Graph Execution, when file with generated data is removed and must be recreated

### How to use source of event during task processing

File, which caused event (considered as source of event) may be used during task processing. i.e. reader/writer components in graph transformations may refer to this file by special placeholders: `${event_file_path}` - path to directory which contains event source `${event_file_name}` - name of event source.

i.e. if event source is: `/home/clover/data/customers.csv`, placeholders will contain: `event_file_path` - `/home/clover/data`, `event_file_name` - `customers.csv`

For "graph execution" task this works only if the graph is not pooled. Thus "keep in pool interval" must be set to 0 (default value).
Chapter 11. WebDAV

Since 3.1

WebDAV API allows user to use standard WebDAV clients for managing sandboxes content.

It allows specifically:

• browsing directory structure
• editing files
• removing files/folders
• renaming files/folders
• creating files/folders
• copying files
• moving files

It is accessible on URL "http://[host]:[port]/clover/webdav".

Although common www browsers can open this URL, most of them are not rich WebDAV clients, thus you can just see list of items, but you cannot browse the directory structure with common www browsers.

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

MS windows

Last distributions of MS Windows (Win XP and later) have native support for WebDAV. Unfortunatelly, it is more or less unreliable, so it is recommended to use some free or commercial WebDAV client. The best WebDAV client we've tested is BitKinex: http://www.bitkinex.com/webdavclient

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

Whereas most of WebDAV clients can work with HTTP Digest Authentication, some of them cannot use HTTP Basic Authentication. So the CloverETL Server WebDAV API uses the Digest Authentication by default. However it may be reconfigured to use HTTP Basic Authentication. Please see the Configuration section for details.

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 12. Simple HTTP API

This API is intended for all HTTP clients (even for the simplest ones - like wget tool). All operations are accessible using http GET method and return plain text. Thus response can be parsed by simple tools. If global security is on (default setting), BASIC HTTP authentication is required. Use CloverETL Server user with proper permissions.

URL has this pattern:

```
http://[domain]:[port]/[context]/[servlet]/[operation]?[param1]=[value1]&[param2]=[value2]...
```

- **Operation help** (p. 59)
- **Operation graph_run** (p. 59)
- **Operation graph_status** (p. 60)
- **Operation graph_kill** (p. 61)
- **Operation server_jobs** (p. 61)
- **Operation sandbox_list** (p. 61)
- **Operation sandbox_content** (p. 61)
- **Operation executions_history** (p. 62)
- **Operation suspend** (p. 63)
- **Operation resume** (p. 63)

### Operation help

**parameters**

no

**returns**

list of possible operations and parameters

**example**

```
http://localhost:8080/clover/request_processor/help
```

### Operation graph_run

**parameters**
Table 12.1. Parameters of graph_run

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphID</td>
<td>yes</td>
<td>-</td>
<td>Text Id, which is unique in specified sandbox. May be file path relative to sandbox root</td>
</tr>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>-</td>
<td>Text ID of sandbox</td>
</tr>
<tr>
<td>runtime config</td>
<td>no</td>
<td>default</td>
<td>Text ID of runtime config for this execution. If not specified, default will be used.</td>
</tr>
<tr>
<td>additional graph parameters</td>
<td>no</td>
<td></td>
<td>Any URL parameter with &quot;param_&quot; prefix is passed to executed graph and may be used in graph XML as placeholder, but without &quot;param_&quot; prefix. i.e. &quot;param_file_name&quot; specified in URL may be used in the graph as ${file_name}. These parameters are resolved only during loading of graph XML, so graph cannot be pooled.</td>
</tr>
</tbody>
</table>

returns

run ID: incremental number, which identifies each execution request

eexample

http://localhost:8080/clover/request_processor/graph_run?graphID=graphDBExecute&sandbox=mva

Operation graph_status

parameters

Table 12.2. Parameters of graph_status

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runID</td>
<td>yes</td>
<td>-</td>
<td>Id of each graph execution</td>
</tr>
<tr>
<td>returnType</td>
<td>no</td>
<td>STATUS</td>
<td>STATUS</td>
</tr>
<tr>
<td>waitForStatus</td>
<td>no</td>
<td>-</td>
<td>Status code which we want to wait for. If it is specified, this operation will wait until graph is in required status.</td>
</tr>
<tr>
<td>waitTimeout</td>
<td>no</td>
<td>0</td>
<td>If waitForStatus is specified, it will wait only specified amount of milliseconds. Default 0 means forever, but it depends on application server configuration. When the specified timeout expires and graph run still isn't in required status, server returns code 408 (Request Timeout). 408 code may be also returned by application server if its HTTP request timeout expires before.</td>
</tr>
</tbody>
</table>

returns

Status of specified graph. It may be number code, text code or complex description in dependence of optional parameter returnType. Description is returned as plain text with pipe as separator, or as XML. Schema describing XML format of the XML response is accessible on CloverETL Server URL: http://[host]:[port]/clover/schemas/executions.xsd In dependence on waitForStatus parameter may return result immediately or wait for specified status.

eexample

http://localhost:8080/clover/request_processor/graph_status ->
Operation graph_kill

parameters

Table 12.3. Parameters of graph_kill

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runID</td>
<td>yes</td>
<td>-</td>
<td>Id of each graph execution</td>
</tr>
<tr>
<td>returnType</td>
<td>no</td>
<td>STATUS</td>
<td>STATUS</td>
</tr>
</tbody>
</table>

returns

Status of specified graph after attempt to kill it. It may be number code, text code or complex description in dependence of optional parameter.

example

http://localhost:8080/clover/request_processor/graph_kill?runID=123456&returnType=DESCRIPTION

Operation server_jobs

parameters

no

returns

List of runID which are currently running.

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 12.4. Parameters of sandbox_content

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>-</td>
<td>text ID of sandbox</td>
</tr>
</tbody>
</table>

returns
List of all elements in specified sandbox. Each element may be specified as file path relative to sandbox root.

eexample

http://localhost:8080/clover/request_processor/sandbox_content?sandbox=mva

Operation executions_history

parameters

Table 12.5. Parameters of executions_history

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>-</td>
<td>text ID of sandbox</td>
</tr>
<tr>
<td>from</td>
<td>no</td>
<td></td>
<td>Lower datetime limit. Operation will return only records after(and equal) this datetime. Format: &quot;yyyy-MM-dd HH:mm&quot; (must be URL encoded)</td>
</tr>
<tr>
<td>to</td>
<td>no</td>
<td></td>
<td>Lower datetime limit. Operation will return only records after(and equal) this datetime. Format: &quot;yyyy-MM-dd HH:mm&quot; (must be URL encoded)</td>
</tr>
<tr>
<td>status</td>
<td>no</td>
<td></td>
<td>Current execution status. Operation will return only records with specified STATUS. Meaningful values are RUNNING</td>
</tr>
<tr>
<td>sandbox</td>
<td>no</td>
<td></td>
<td>Sandbox code. Operation will return only records for graphs from specified sandbox.</td>
</tr>
<tr>
<td>graphId</td>
<td>no</td>
<td></td>
<td>Text Id, which is unique in specified sandbox. File path relative to sandbox root</td>
</tr>
<tr>
<td>orderBy</td>
<td>no</td>
<td></td>
<td>Attribute for list ordering. Possible values: id</td>
</tr>
<tr>
<td>orderDescend</td>
<td>no</td>
<td>true</td>
<td>Switch which specifies ascending or descending ordering. If it is true (which is default), ordering is descending.</td>
</tr>
<tr>
<td>returnType</td>
<td>no</td>
<td>IDs</td>
<td>Possible values are: IDs</td>
</tr>
<tr>
<td>index</td>
<td>no</td>
<td>0</td>
<td>Index of the first returned records in whole record set. (starting from</td>
</tr>
<tr>
<td>records</td>
<td>no</td>
<td>infinite</td>
<td>Max amount of returned records.</td>
</tr>
</tbody>
</table>

returns
List of executions according to filter criteria.

For returnType==IDs returns simple list of runIDs (with new line delimiter).

For returnType==DESCRIPTION returns complex response which describes current status of selected executions, their phases, nodes and ports.
Chapter 12. Simple HTTP API

```plaintext
execution|runID|status|username|sandbox|graphID|startedDatetime|finishedDatetime|clusterNode|graphPhase|index|executeTimeInMilis
node|nodeID|status|totalCpuTime|totalUserTime|cpuUsage|peakCpuUsage|userUsage|peakUserUsage
port|portType|index|avgBytes|avgRows|peakBytes|peakRows|totalBytes|totalRows
```

**example of request**

```plaintext
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**

```plaintext
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|130|10
node|TRASH0|FINISHED_OK|0|0.0|0.0|0.0|0.0|0.0
port|Input|0|0|0|0|130|10
node|SPEED_LIMITER0|FINISHED_OK|0|0|0.0|0.0|0.0|0.0
port|Input|0|0|0|0|130|10
port|Output|0|0|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|130|10
node|TRASH0|RUNNING|0|0|0.0|0.0|0.0|0.0
port|Input|0|5|0|5|52|4
node|SPEED_LIMITER0|RUNNING|0|0|0.0|0.0|0.0|0.0
port|Input|0|0|0|0|130|10
port|Output|0|5|0|5|52|4
```

For returnType==DESCRIPTION.XML returns complex data structure describing one or more selected executions in XML format. Schema describing XML format of the XML response is accessible on CloverETL Server URL: http://[host][:port]/clover/schemas/executions.xsd

**Operation suspend**

Suspend server or sandbox(if specified). Suspension means, that no graphs may me executed on suspended server/sandbox.

**parameters**

**Table 12.6. Parameters of suspend**

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sandbox</td>
<td>no</td>
<td>-</td>
<td>Text ID of sandbox to suspend. If not specified, it suspends whole server.</td>
</tr>
<tr>
<td>atonce</td>
<td>no</td>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

**returns**

Result message

**Operation resume**

**parameters**
Table 12.7. Parameters of resume

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sandbox</td>
<td>no</td>
<td>-</td>
<td>Text Id of sandbox to resume. If not specified, server will be resumed.</td>
</tr>
</tbody>
</table>

returns

Result message
Chapter 13. JMX mBean

CloverETL Server JMX mBean is API, which is useful for monitoring of CloverETL Server's internal status.

MBean is registered with name:

```java
com.cloveretl.server.api.jmx:name=cloverServerJmxMBean
```

JMX configuration

Application's JMX MBeans aren't accessible outside of JVM by default. It needs some changes in application server configuration to make them accessible.

This section describes how to configure JMX Connector for development and testing. Thus authentication may be disabled. For production deployment authentication should be enabled. Please refer further documentation to see how to achieve this. i.e. [http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html#auth](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. 65)
- How to configure JMX on Glassfish (p. 66)
- Websphere 7 (p. 67)
- Possible problems (p. 69)

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`

On UNIX like OS set environment variable CATALINA_OPTS i.e. like this:

```bash
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"
```

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:

```bash
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.ssl=false
set CATALINA_OPTS=%CATALINA_OPTS% -Dcom.sun.management.jmxremote.authenticate=false
```

65
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 Glasfish admin console (by default accessible on [http://localhost:4848](http://localhost:4848) with admin/adminadmin as user/password)

Go to section "Configuration" > "Admin Service" > "system" and set attributes like this:

![Glassfish JMX connector](image)

**Figure 13.1. 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:
Figure 13.2. Websphere configuration

Websphere 6

URL for connecting to JMX server is:

```
service:jmx:rmi:///jndi/JMXConnector
```

Following system properties need to be set:

```
org.omg.CORBA.ORBClass=com.ibm.CORBA.iioORB
java.naming.provider.url=corbaloc:iiop:[host]:[port]/WsnAdminNameService
```

If you have a default Websphere installation, the JNDI port number will likely be 2809, 2810,... 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 RMIC0026I: The RMI Connector is available at port 2810
```

You will also need to set on the classpath following jar files from Websphere home directory:

```
/runtimes/com.ibm.ws.admin.client_6.1.0.jar
/runtimes/runtimes/com.ibm.ws.webservices.client_6.1.0.jar
/java/jre/lib/ibmorb.jar
```

Websphere 7

URL for connecting to JMX server is:

```
service:jmx:iiop://[host]:[port]/jndi/JMXConnector
```
where \textit{host} is the host name you are connecting to and \textit{port} is RMI port number. If you have a default Websphere installation, the JNDI port number will likely be 2809, 2810, ... 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 RMIC00026I: The RMI Connector is available at port 2810
```

### How to configure JMX on Websphere7

Websphere does not require any special configuration, but the clover MBean is registered with the name, that depends on application server configuration:

```jmx
com.cloveretl.server.api.jmx:cell=[cellName],name=cloverServerJmxMBean,node=[nodeName],
 process=[instanceName]
```

**Figure 13.3. Websphere7 configuration**

URL for connecting to JMX server is:

```
service:jmx:iiop://[host]:[port]/jndi/JMXConnector
```

where \textit{host} is the host name you are connecting to and \textit{port} is RMI port number. If you have a default Websphere installation, the JNDI port number will likely be 2809, 2810, ... 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 RMIC00026I: The RMI Connector is available at port 2810
```

You will also need to set on the classpath following jar files from Websphere home directory:

```bash
/run times/com.ibm.ws.admin.client_7.0.0.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

Because JMX is stateless communication, all operations have at least two parameters: user and password.

List of operations is the following:

- Operation getServerJobs (p. 69)
- Operation executeGraph (p. 69)
- Operation killGraph (p. 70)
- Operation graphStatus (p. 70)
- Operation suspendServer (p. 71)
- Operation resumeServer (p. 71)
- Operation suspendServerSandbox (p. 71)
- Operation resumeServerSandbox (p. 72)
- Operation getGraphExecutionMBeanName (p. 72)

**Operation getServerJobs**

**parameters**

Table 13.1. Parameters of getServerJobs

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

**returns**

IDs of running jobs(graphs). IDs may be used as parameters of another operations.

**Operation executeGraph**

Since: 1.2.1

Executes specified graph. Only user, which has permission to execute the graph may call operation executeGraph(). Otherwise CloverSecurityException is thrown.

**parameters**
Table 13.2. Parameters of executeGraph

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>String</td>
<td>Text ID of sandbox which contains graph to execute.</td>
</tr>
<tr>
<td>graphId</td>
<td>yes</td>
<td>String</td>
<td>Text ID of graph. It is path to graph file relative to sandbox root. Only forward slashes may be used.</td>
</tr>
</tbody>
</table>

returns

Result runID of execution or throws an exception.

Operation killGraph

Kill running graph. Only user, which has permission to execute the graph may call operation killGraph() to kill it. Otherwise CloverSecurityException is thrown.

parameters

Table 13.3. Parameters of killGraph

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>runID</td>
<td>yes</td>
<td>long</td>
<td>Run ID, which may be obtained i.e. by getRunningJobs() operation.</td>
</tr>
</tbody>
</table>

returns

Result status of killed graph. If it is successfully killed, status should be ABORTED.

Operation graphStatus

Since: 1.2.1

Returns current status of specified execution.

parameters

Table 13.4. Parameters of graphStatus

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>runID</td>
<td>yes</td>
<td>long</td>
<td>Run ID, which is returned by executeGraph method or which may be obtained i.e. by getRunningJobs() operation.</td>
</tr>
</tbody>
</table>

returns

Result status of specified graph execution. Possible values are: FINISHED_OK | RUNNING | N_A | ERROR | ABORTED | READY
Chapter 13. JMX mBean

**Operation suspendServer**

Suspends server. Suspended server means that no graph may be executed. All attempts to execute graph will fail. See resumeServer operation. Only administrator can call this operation. Otherwise CloverSecurityException is thrown.

parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>atOnce</td>
<td>yes</td>
<td>boolean</td>
<td>If this param is set to true, running graphs from suspended server are aborted. Otherwise it can run until it is finished in common way.</td>
</tr>
</tbody>
</table>

returns

void

**Operation resumeServer**

Resumes suspended server. Only administrator can call this operation. Otherwise CloverSecurityException is thrown. See suspendServer() operation.

parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

returns

void

**Operation suspendServerSandbox**

Suspends specified sandbox. Suspended sandbox means that no graph from sandbox may be executed. All attempts to execute graph will fail. See resumeServerSandbox operation. Only administrator can call this operation. Otherwise CloverSecurityException is thrown.

parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>String</td>
<td>Text ID of sandbox to suspend</td>
</tr>
<tr>
<td>atOnce</td>
<td>yes</td>
<td>boolean</td>
<td>If this param is set to true, running graphs from suspended sandbox are aborted. Otherwise it can run until it is finished in common way.</td>
</tr>
</tbody>
</table>
returns
void

**Operation resumeServerSandbox**

Resumes suspended sandbox. Only administrator can call this operation. Otherwise CloverSecurityException is thrown. See suspendServerSandbox() operation.

parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>sandbox</td>
<td>yes</td>
<td>String</td>
<td>Text ID of sandbox to suspend</td>
</tr>
</tbody>
</table>

returns
void

**Operation getGraphExecutionMBeanName**

Returns MBean name of running graph. It may be used for direct monitoring of the transformation. However in cluster environment, MBean is accessible only on the node, which runs graph.

parameters

<table>
<thead>
<tr>
<th>parameter name</th>
<th>mandatory</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>yes</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>runID</td>
<td>yes</td>
<td>long</td>
<td>Run ID, which may be obtained i.e. by getRunningJobs() operation.</td>
</tr>
</tbody>
</table>
Chapter 14. SOAP WebService API

CloverETL Server SOAP Web Service is API, which allows its clients to manipulate with content of the sandboxes, to monitor status of executed graphs and more.

Service is accessible on URL:

\[
\text{http://[host]:[port]/clover/webservice}
\]

Service descriptor is accessible on URL:

\[
\text{http://[host]:[port]/clover/webservice?wsdl}
\]

Protocol HTTP can be changed to secured HTTPS according to web server configuration.

SOAP WS Client

Exposed service is implemented with the most common binding style "document/literal", which is widely supported by libraries in various programming languages.

To create client for this API, only WSDL document (see the URL above) is needed together with some development tools according to your programming language and development environments.

JavaDoc of WebService interface with all related classes is accessible in the running CloverETL Server instance on URL http://[host]:[port]/[contextPath]/javadoc-ws/index.html

If the web server has HTTPS connector configured, also the client must meet the security requirements according to web server configuration. i.e. client trust + key stores configured properly

SOAP WS API authentication/authorization

Since exposed service is stateless, authentication "sessionToken" has to be passed as parameter to each operation. Client can obtain authentication sessionToken by calling "login" operation.
Chapter 15. Launch Service

The Launch Service provides users with convenient way of remotely executing the CloverETL graphs via a simple web-based interface which can be customized to fit the needs of the users.

The Launch Services can be used with any browser and therefore do not require users to install any software. This allows for convenient control of the graph execution which can be easily tied to external tools if necessary (requests can be sent from custom applications as well).

Launch Service Overview

The architecture of Launch Service is relatively simple and follows the basic design of multi-tiered applications utilizing the browser.

![Figure 15.1. Launch Service Overview](image)

The basic usage scenario of client form page is simple:

1. User opens the Launch Service web page in his browser
2. User enters the parameters of the graph (if required)
3. The data is submitted to the CloverETL Server. This is the moment the service is actually called.
4. All the results (including error messages or logs) are sent back to the user and displayed in the browser. The logs are also available for inspection via CloverETL Server GUI.

The Launch Service web pages which are presented to users can be fully replaced by client's web application or by third party application which calls CloverETL Launch Service by HTTP request. This allows full customization of the outside appearance of the web - for example, it can be a simple web form which communicates with users in the terminology they are familiar with.
Chapter 15. Launch Service

Deploying Graph in Launch Service

To enable users to access the specific graph via Launch Service, several steps have to be taken:

1. The graph has to be designed to allow its parameters to be passed via dictionary.
2. The graph has to be configured in CloverETL Server in Launch Service section.
3. The form which will submit the data to Launch Service has to be written.

Overall the deployment of the graph to the Launch Service is not much more complex compared to the regular graph development process. In following chapters all the steps will be described in more detail alongside some basic examples.

Designing the Graphs for Launch Service

To use the graphs from Launch Service, the Launch Service requires the graph to use dictionary when parameters have to be passed to the graph. Dictionary is a data storage associated with each run of the graph in CloverETL. For more details about the dictionary see section "Dictionary" in CloverETL Designer docs.

To use the Dictionary from the Launch Service, the graph author is required to specify the entries of the dictionary in graph's XML source file. For more details about the Dictionary XML element see section "Dictionary" in CloverETL Designer docs.

Apart from the use of the dictionary, the Launch Service does not impose any other restriction on the graphs it should run. The graphs can therefore use all the facilities provided by the CloverETL engine.

Configuring the Graph in CloverETL Server web GUI

To notify the Launch Service about the graphs that will be available via its interface, the Launch Service has to be properly configured via CloverETL Server GUI.

Launch Service uses launch configurations to store the details about how each graph can be run. Each launch configuration contains full description of the graph's parameters, how they are mapped to the parameters passed from the web interface and so on.

Each launch configuration is identified by its name, user and group restriction. Several configurations with the same name can be created as long as they differ in their user or group restrictions.

Use restrictions can be used to launch different graphs for different users even though they use the same launch configuration (for example, the developers may want to use debug version of the graph while the end customers will want to use the production graph). The user restriction can also be used to prohibit certain users from executing the launch configuration.

Similarly, the group restriction can be used to differentiate graphs based on the group membership of the user which runs the launch configuration.

When the configuration is launched, the correct configuration is picked based on the configuration name, user specification and group specification. If multiple configuration 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

New launch configurations can be added by clicking on New launch configuration link on the Launch Services tab in CloverETL Server GUI.
Figure 15.2. Launch Service section

After the configuration has been created it will appear in the table on the left side among the other existing configuration. Before using the configuration user will have to add parameter mapping. To add parameter mappings click on the detail link for the newly created configuration. The details will be displayed on the right side of the window in a simple table:

![Figure 15.3. The Basic Info tab](image)

The Basic Info tab shows the basic details about the launch configuration. These can be modified in the Edit Configuration tab:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>New/Mountains</td>
</tr>
<tr>
<td>Description</td>
<td>Filter mountains based on height</td>
</tr>
<tr>
<td>Group</td>
<td>default</td>
</tr>
<tr>
<td>User</td>
<td>graph/graphMountainsXLS_grf</td>
</tr>
<tr>
<td>sandbox</td>
<td>false</td>
</tr>
<tr>
<td>Graph</td>
<td>false</td>
</tr>
<tr>
<td>Save run record</td>
<td>false</td>
</tr>
<tr>
<td>Display error message detail</td>
<td>false</td>
</tr>
</tbody>
</table>
Chapter 15. Launch Service

Figure 15.4. Edit Configuration tab

Following fields can be modified:

- **Name** - is the name under which the configuration will be accessible from web.

- **Description** - the description of the configuration.

- **Group** - restricts the configuration to specific group of users.

- **User** - restricts the configuration to specified user.

- **Sandbox** - selects the CloverETL Sandbox in which the configuration will be launched.

- **Graph** - selects the graph to run when the configuration is launched.

- **Save run record** - if checked, the details about the launch configuration will be visible in Execution History in the CloverETL Server GUI. If unchecked, the graph executions will not be logged and will not be displayed in the Execution History.

- **Display error message detail** - if checked, detailed error messages will be displayed in case the launch fails. If unchecked, only simpler messages will be displayed to the user.

Finally, the tab Edit Parameters can be used to configure parameter mappings for the launch configuration. The mappings are required for the Launch Service to be able to correctly assign parameters values based on the values sent in the launch request.

Figure 15.5. Edit Parameters tab

To add new parameter mapping click on the New property link. Each property required by the graph has to be created (internal graph properties do not need mappings).
Following fields are available for each property:

- **Name** - the name of the property in the graph's dictionary.

- **Request parameter** - the name of the parameter as specified in the launch request generated by the request page. This name can be different than the name used in graph's dictionary.

- **Parameter required** - if checked the parameter is mandatory and error will be reported if it is omitted.

- **Pass to graph** - if checked the parameter will be also passed to graph among the additional parameters as well as in the dictionary. In such case, the parameter can also be referenced as ${PropertyName} in the graph's XML file. Since the additional parameters are resolved when the XML file is parsed, the graphs which use this method cannot be pooled.

- **Default value** - is the default value which will be applied in case the parameter is omitted in the launch request.

To create the new mapping, click on the Create button after all the fields have been filled. After the mapping is created, it will be displayed in the list of existing mappings. It can be later edited or deleted by clicking on appropriate links.

---

**Sending the Data to Launch Service**

To launch the graph which has been configured for use with Launch Service, the user has to send a launch request. The launch request can be sent via HTTP GET or POST methods. A launch request is simply an URL which contains the values of all parameters that should be passed to the graph. The request URL is composed of several parts:

```
[Clover Context]/launch/[Configuration name]?[Parameters]
```
• [Clover Context] is the URL to the context in which the CloverETL is running. Usually this is the full URL to 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 which has been specified when the configuration has been created. In our example, this would be set to NewMountains (distinction between upper- and lower-case is important).

• [Parameters] is the list of parameters the configuration requires in the format used for example by PHP. Therefore the parameter list is a list of name-value pairs separated by "&" character. Each name-value pair is specified as [name]=[value] where value has to be properly encoded according to RFC 1738 to make sure URL is valid.

Based on the above, the full URL of launch request for our example with mountains may be 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 graph's run terminates, the results are sent back from the engine to the server and finally to the user. The output is partially defined in the dictionary which is declared in the graph's XML file. The dictionary can mark selected parameters as output parameters. All the output parameters are sent to the user after the graph execution is finished.

Depending on the number of output parameters, the following output is sent to user:

• **No output parameters** - only summary page is displayed to the user. The format of the summary page cannot be customized. The page will contain details like when the graph was started, when it finished, user name and so on.

• **One output parameter** - in this case the output is sent to the user with its content type defined by the property type in the dictionary.

• **Multiple output parameters** - in this case each output parameter is sent to user a part of multipart response. The content type of the response is either multipart/related or multipart/x-mixed-replace depending on the target browser (the browser detection is of course fully automatic). The multipart/related type is used for browsers based on Microsoft Internet Explorer, the multipart/x-mixed-replace is sent to browsers based on Gecko or Webkit.

Launch requests are recorded in the log files in directory specified by launch.log.dir property in 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 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 case 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.
Chapter 16. Configuration

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

Config Sources and Their Priorities

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

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

Context Parameters (Available on Apache Tomcat)

Some application servers allows to set context parameters without modification of WAR file. This way of configuration is possible and recommended for Tomcat.

Example for Apache Tomcat

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

You can specify property by adding this element:

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

Environment Properties

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

Properties File on default Location

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

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

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

Properties File on specified Location

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

Modification of Context Parameters in web.xml

Unzip clover.war and modify file WEB-INF/web.xml, add this code:
This way isn't recommended, but it may take place when none of above ways is possible.

**Priorities of config Sources**

Configuration sources have these priorities:

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

**Examples of DB Connection Configuration**

Configuration of DB connection is optional. Embedded Apache Derby DB is used by default and it is sufficient for evaluation, however configuration of external DB connection is strongly recommended for production deployment. It is possible to specify common JDBC DB connection attributes (URL, username, password) or JNDI location of DB DataSource.

Configurations and their changes may be as follows:

- **Upgrade of DB schema** (p. 82)
- **Embedded Apache Derby** (p. 83)
- **MySQL** (p. 83)
- **DB2** (p. 84)
- **Oracle** (p. 86)
- **MS SQL** (p. 86)
- **Postgre SQL** (p. 87)
- **JNDI DB DataSource** (p. 87)

**Upgrade of DB schema**

If you replace older version of CloverETL Server by new one above the same DB, there may be some changes in DB schema. Since CloverETL Server version 1.2, DB patches above existing DB schema are done automatically, during first startup. However If you are upgrading from DB schema of version 1.1. you will have to preset this feature by these SQL updates:

```sql
CREATE TABLE sys_schema_patches (patch varchar(256) unique not null, applied timestamp);
```
Chapter 16. Configuration

```
INSERT INTO sys_schema_patches (patch,applied) values ('0000_create.sql', null );
```

Do not execute it above empty DB! It is intended only for upgrading from existing DB schema of 1.1. version.

### Embedded Apache Derby

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

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

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

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```
<Parameter name="jdbc.driverClassName" value="org.apache.derby.jdbc.EmbeddedDriver" override="false" />
<Parameter name="jdbc.url" value="jdbc:derby:databases/cloverDb;create=true" override="false" />
<Parameter name="jdbc.username" value="" override="false" />
<Parameter name="jdbc.password" value="" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.DerbyDialect" override="false" />
```

Or If you use properties file for configuration:

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

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

### MySQL

CloverETL Server requires MySql 5.x

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```
<Parameter name="jdbc.driverClassName" value="com.mysql.jdbc.Driver" override="false" />
<Parameter name="jdbc.url" value="jdbc:mysql://127.0.0.1:3306/clover?useUnicode=true&amp;characterEncoding=utf8" override="false" />
<Parameter name="jdbc.username" value="root" override="false" />
<Parameter name="jdbc.password" value="" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.MySQLDialect" override="false" />
```

Or If you use properties file for configuration:

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

Since 3.0 JDBC driver isn't included in CloverETL Server web archive, thus it must be added to the application server classpath.
Create DB with proper charset, like this:

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

**DB2**

**DB2 on Linux/Windows**

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```
<Parameter name="jdbc.driverClassName" value="com.ibm.db2.jcc.DB2Driver" override="false" />
<Parameter name="jdbc.url" value="jdbc:db2://localhost:50000/clover" override="false"/>
<Parameter name="jdbc.username" value="usr" override="false"/>
<Parameter name="jdbc.password" value="pwd" override="false"/>
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.DB2Dialect" override="false"/>
```

Or If you use properties file for configuration:

```
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
```

**Possible problems**

**Wrong pagesize**

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

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

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

SQLERRMC contains suitable value for `PAGESIZE`.

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

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

**The table is in the reorg pending state**

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

Error message for this issue may look like this:

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

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

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

```sql
reorg table run_record
```

DB2 console output should look like this:

```
db2 => connect to clover1
Database Connection Information
Database server        = DB2/LINUX 9.7.0
SQL authorization ID   = DB2INST2
Local database alias   = CLOVER1
db2 => reorg table run_record
DB20000I  The REORG command completed successfully.
db2 => disconnect clover1
DB20000I  The SQL DISCONNECT command completed successfully.
```

"clover1" is DB name

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

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

**DB2 on AS/400**

The connection on AS/400 might be slightly different.

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```xml
<Parameter name="jdbc.driverClassName" value="com.ibm.as400.access.AS400JDBCDriver" override="false" />
<Parameter name="jdbc.url" value="jdbc:as400://localhost/cloversrv;date format=iso" override="false" />
<Parameter name="jdbc.username" value="javlin" override="false" />
<Parameter name="jdbc.password" value="clover" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.DB2400Dialect" override="false" />
```

Or If you use properties file for configuration:

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

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

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

You can create schema in AS/400 console:
• execute command STRSQL *(SQL console)*

• execute CREATE COLLECTION cloversrv IN ASP 1

• `cloversrv` is the name of the DB schema and it may be at most 10 characters long

Proper JDBC driver must be in the application server classpath.

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

Use `jt400ntv.jar` JDBC driver.

Do not forget to add jar with JDBC driver to the Tomcat classpath.

**Oracle**

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```xml
<Parameter name="jdbc.driverClassName" value="oracle.jdbc.OracleDriver" override="false" />
<Parameter name="jdbc.url" value="jdbc:oracle:thin:@host:1521:db" override="false" />
<Parameter name="jdbc.username" value="user" override="false" />
<Parameter name="jdbc.password" value="pass" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.Oracle9Dialect" override="false" />
```

Or If you use properties file for configuration:

```properties
jdbc.driverClassName=oracle.jdbc.OracleDriver
jdbc.url=jdbc:oracle:thin:@host:1521:db
jdbc.username=user
jdbc.password=pass
jdbc.dialect=org.hibernate.dialect.Oracle9Dialect
```

Do not forget to add jar with JDBC driver to the application server classpath.

Since CloverETL Server version 1.2.1, dialect `org.hibernate.dialect.Oracle10gDialect` is no longer available. Please use `org.hibernate.dialect.Oracle9Dialect` instead.

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

```sql
CONNECT
CREATE SESSION
CREATE/ALTER/DROP TABLE
CREATE/ALTER/DROP SEQUENCE
QUOTA UNLIMITED ON <user_tablespace>;
QUOTA UNLIMITED ON <temp_tablespace>;
```

**MS SQL**

Ms SQL requires configuration of DB server.

• Allowing of TCP/IP connection:

• execute tool *SQL Server Configuration Manager*

• go to *Client protocols*

• switch on TCP/IP (default port is 1433)
Chapter 16. Configuration

- execute tool SQL Server Management Studio
- go to Databases and create DB clover
- go to Security/Logins and create user and assign this user as owner of DB clover
- go to Security and check SQL server and Windows authentication mode

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```xml
<Parameter name="jdbc.driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" override="false" />
<Parameter name="jdbc.url" value="jdbc:sqlserver://localhost:1433;databaseName=clover" override="false" />
<Parameter name="jdbc.username" value="user" override="false" />
<Parameter name="jdbc.password" value="pass" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.SybaseDialect" override="false" />
```

Or If you use properties file for configuration:

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

Do not forget to add jar with JDBC driver to the Tomcat classpath.

**PostgreSQL**

For modification Tomcat context params, add to context config file (and modify according to your credentials):

```xml
<Parameter name="jdbc.driverClassName" value="org.postgresql.Driver" override="false" />
<Parameter name="jdbc.url" value="jdbc:postgresql://localhost/clover?charSet=UTF-8" override="false" />
<Parameter name="jdbc.username" value="postgres" override="false" />
<Parameter name="jdbc.password" value="" override="false" />
<Parameter name="jdbc.dialect" value="org.hibernate.dialect.PostgreSQLDialect" override="false" />
```

Or If you use properties file for configuration:

```properties
jdbc.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
driver.url=jdbc:postgresql://localhost/clover?charSet=UTF-8
driver.username=postgres
driver.password=
driver.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Do not forget to add jar with JDBC driver to the Tomcat classpath.

**JNDI DB DataSource**

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

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

Above parameters may be set in the same ways as other params (in properties file or Tomcat context file)
Example of DataSource configuration in Apache Tomcat. Add following code to context file.

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

### List of Properties

#### Table 16.1. General configuration

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>config.file</td>
<td>location of CloverETL Server configuration file</td>
<td>[working_dir]/cloverServer.properties</td>
</tr>
<tr>
<td>license.file</td>
<td>location of CloverETL Server licence file (license.dat)</td>
<td></td>
</tr>
<tr>
<td>engine.config.file</td>
<td>location of CloverETL engine configuration properties file</td>
<td>properties file packed with CloverETL</td>
</tr>
<tr>
<td>private.properties</td>
<td>List of server properties which are used only by CloverETL Server code. So these properties are not accessible outside of the ServerFacade. By default there are all properties which may contain password in the list. Basically it means, that their values are not visible for web GUI users. Values are replaced by single star &quot;*&quot;. Changes in this list may cause unexpected behavior of some server API.</td>
<td>jdbc.password,executor.password,security.ldap.password,clover.smtp.password</td>
</tr>
<tr>
<td>engine.plugins.src</td>
<td>This property may contain absolute path to some &quot;source&quot; of additional CloverETL engine plugins. These plugins are not a substitute for plugins packed in WAR. &quot;Source&quot; may be directory or zip file. Both directory and zip must contain subdirectory for each plugin. Changes in the directory or the ZIP file apply only when the server is restarted. For details see section &quot;Extensibility - engine plugins&quot;.</td>
<td>empty</td>
</tr>
<tr>
<td>key</td>
<td>description</td>
<td>default</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>datasource.type</td>
<td>Set this explicitly to JNDI if you need CloverETL Server to connect to DB using JNDI datasource. In such case, parameters &quot;datasource.jndiName&quot; and &quot;jdbc.dialect&quot; must be set properly. Possible values: JNDI</td>
<td>JDBC</td>
</tr>
<tr>
<td>datasource.jndiName</td>
<td>JNDI location of DB DataSource. It is applied only if &quot;datasource.type&quot; is set to &quot;JNDI&quot;.</td>
<td>java:comp/env/jdbc/clover_server</td>
</tr>
<tr>
<td>jdbc.driverClassName</td>
<td>class name for jdbc driver name</td>
<td></td>
</tr>
<tr>
<td>jdbc.url</td>
<td>jdbc url used by CloverETL Server to store data</td>
<td></td>
</tr>
<tr>
<td>jdbc.username</td>
<td>jdbc database user name</td>
<td></td>
</tr>
<tr>
<td>jdbc.password</td>
<td>jdbc database user name</td>
<td></td>
</tr>
<tr>
<td>jdbc.dialect</td>
<td>hibernate dialect to use in ORM</td>
<td></td>
</tr>
<tr>
<td>quartz.driverDelegateClass</td>
<td>SQL dialect for quartz. Value is automatically derived from &quot;jdbc.dialect&quot; property value.</td>
<td></td>
</tr>
<tr>
<td>security_enabled</td>
<td>true</td>
<td>false If it is set to false, then no authentication is required and anyone has admin privileges.</td>
</tr>
<tr>
<td>security.default_domain</td>
<td>Domain which all new users are included in. Stored in user's record in the database. Shouldn't be changed unless the &quot;clover&quot; must be white-labelled.</td>
<td>clover</td>
</tr>
<tr>
<td>security.basic_authentication.features_list</td>
<td>Semi-colon separated list of features which are accessible using HTTP and which should be protected by Basic HTTP Authentication. Each feature is specified by its servlet path.</td>
<td>/request_processor:/simpleHttpApi;/launch;/launchIt;/downloadStorage;/downloadFile;/uploadSandboxFile;/downloadLog</td>
</tr>
<tr>
<td>security.basic_authentication.realm</td>
<td>Realm string for HTTP Basic Authentication.</td>
<td>CloverETL Server</td>
</tr>
<tr>
<td>key</td>
<td>description</td>
<td>default</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>security.digest_authentication.features_list</td>
<td>Semi-colon separated list of features which are accessible using HTTP and which should be protected by HTTP Digest Authentication. Each feature is specified by its servlet path. Please keep in mind, that HTTP Digest Authentication is feature added to the version 3.1. If you upgraded your older CloverETL Server distribution, users created before the upgrade cannot use the HTTP Digest Authentication until they reset their passwords. So when they reset their passwords (or the admin does it for them), they can use Digest Authentication as well as new users.</td>
<td>/webdav</td>
</tr>
<tr>
<td>security.digest_authentication.realm</td>
<td>Realm string for HTTP Digest Authentication. If it is changed, all users have to reset their passwords, otherwise they won't be able to access to the server features protected by HTTP digest Authentication.</td>
<td>CloverETL Server</td>
</tr>
<tr>
<td>security.digest_authentication.nonce_validity</td>
<td>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.</td>
<td>300</td>
</tr>
<tr>
<td>clover.event.fileCheckMinInterval</td>
<td>Interval of file checkes (in milliseconds) See Chapter 10, [File event listeners](p. 54) for details.</td>
<td>1000</td>
</tr>
<tr>
<td>clover.smtp.host</td>
<td>SMTP server hostname or IP address</td>
<td></td>
</tr>
<tr>
<td>clover.smtp.port</td>
<td>SMTP server port</td>
<td></td>
</tr>
<tr>
<td>clover.smtp.authentication</td>
<td>true/false If it is false, username and password are ignored</td>
<td></td>
</tr>
<tr>
<td>clover.smtp.username</td>
<td>SMTP server username</td>
<td></td>
</tr>
<tr>
<td>clover.smtp.password</td>
<td>SMTP server password</td>
<td></td>
</tr>
<tr>
<td>key</td>
<td>description</td>
<td>default</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>logging.project_name</td>
<td>used in log messages where it is necessary to name the product name</td>
<td>CloverETL</td>
</tr>
<tr>
<td>logging.default_subdir</td>
<td>name of default subdirectory for all server logs; it is relative to the path specified by system property &quot;java.io.tmpdir&quot;. Don't specify absolute path, use properties which are intended for absolute path.</td>
<td>cloverlogs</td>
</tr>
<tr>
<td>launch.log.dir</td>
<td>Location, where server should store launch requests logs. See Launch Services section for details.</td>
<td>${java.io.tmpdir}/[logging.default_subdir]/launch where ${java.io.tmpdir} is system property</td>
</tr>
<tr>
<td>graph.logs_path</td>
<td>Location, where server should store Graph run logs. See Logging section for details.</td>
<td>${java.io.tmpdir}/[logging.default_subdir]/graph where ${java.io.tmpdir} is system property</td>
</tr>
<tr>
<td>temp.default_subdir</td>
<td>Name of default subdirectory for server tmp files; it is relative to the path specified by system property &quot;java.io.tmpdir&quot;.</td>
<td>clovertmp</td>
</tr>
<tr>
<td>graph.debug_path</td>
<td>Location, where server should store Graph debug info.</td>
<td>${java.io.tmpdir}/[temp.default_subdir]/debug where ${java.io.tmpdir} is system property</td>
</tr>
<tr>
<td>graph.dictionary_path</td>
<td>Location, where server should store graph dictionary temporary files.</td>
<td>${java.io.tmpdir}/[temp.default_subdir]/dictionary where ${java.io.tmpdir} is system property</td>
</tr>
<tr>
<td>graph.pass_event_params_to_graph_in_old_style</td>
<td>3.0. It is switch for backwards compatibility of passing parameters to the graph executed by graph event. In version prior to 3.0 all params has been passed to executed graph. Since 3.0 just specified parameters are passed. Please see Task - Execution of Graph (p. 35) for details.</td>
<td>false</td>
</tr>
<tr>
<td>threadManager.pool.corePoolSize</td>
<td>Number of threads which are always active (running or idling). Related to thread pool for processing server events.</td>
<td>4</td>
</tr>
<tr>
<td>key</td>
<td>description</td>
<td>default</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>threadManager.pool.queueCapacity</td>
<td>Max size of the queue (FIFO) which contains tasks waiting for thread. Related to thread pool for processing server events. It means, that there won't be more then &quot;queueCapacity&quot; waiting tasks. i.e. queueCapacity=0 - no waiting tasks, each task is immediately executed in available thread or in new thread. queueCapacity=1024 - up to 1024 tasks may be waiting in the queue for available thread from &quot;corePoolSize&quot;.</td>
<td>12</td>
</tr>
<tr>
<td>threadManager.pool.maxPoolSize</td>
<td>Max number of active threads. If no thread from core pool is available and queue capacity is exceeded, pool creates new threads up to &quot;maxPoolSize&quot; threads. If there are more concurrent tasks then maxPoolSize, thread manager refuses to execute it. Thus keep queueCapacity or maxPoolSize big enough.</td>
<td>1024</td>
</tr>
<tr>
<td>task.archivator.batch_size</td>
<td>Max number of records deleted in one batch. It is used for deleting of archived run records.</td>
<td>50</td>
</tr>
<tr>
<td>launch.http_header_prefix</td>
<td>Prefix of HTTP headers added by launch services to the HTTP response.</td>
<td>X-cloveretl</td>
</tr>
<tr>
<td>task.archivator.archive_file_prefix</td>
<td>Prefix of archive files created by archivator.</td>
<td>cloverArchive_</td>
</tr>
<tr>
<td>license.context_names</td>
<td>Comma separated list of web-app contexts which may contain license. Each of them has to start with slash! Works only on Apache Tomcat.</td>
<td>/clover-license,/clover_license</td>
</tr>
<tr>
<td>license.display_header</td>
<td>Switch which specifies whether display license header in server web GUI or not.</td>
<td>false</td>
</tr>
</tbody>
</table>
Table 16.2. Defaults for graph execution configuration - see section Graph config properties for details

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>executor.tracking_interval</td>
<td>Interval in milliseconds for scanning current status of running graph. The shorter interval, the bigger log file.</td>
<td>2000</td>
</tr>
<tr>
<td>executor.log_level</td>
<td>Log level of graph runs. TRACE</td>
<td>DEBUG</td>
</tr>
<tr>
<td>executor.max_running_concurrently</td>
<td>Amount of graph instances which may exist(or run) concurrently. 0 means no limits</td>
<td>0</td>
</tr>
<tr>
<td>executor.max_graph_instance_age</td>
<td>Interval in milliseconds. Specifies how long graph instance can be idling before it is released from memory. 0 means no limits. This property has been renamed since 2.8. Original name was executor.maxGraphInstanceAge</td>
<td>0</td>
</tr>
<tr>
<td>executor.classpath</td>
<td>Classpath for transformation/processor classes used in the graph. Directory [sandbox_root]/trans does not have to be listed here, since it is automatically added to graph run classpath.</td>
<td></td>
</tr>
<tr>
<td>executor.skip_check_config</td>
<td>Disables check of graph configuration. Increases performance of graph execution, however may be useful during graph development.</td>
<td>true</td>
</tr>
<tr>
<td>executor.password</td>
<td>Password for decoding of encoded DB connection passwords.</td>
<td></td>
</tr>
<tr>
<td>executor.verbose_mode</td>
<td>If true, more descriptive logs of graph runs are generated.</td>
<td>true</td>
</tr>
<tr>
<td>executor.use_jmx</td>
<td>If true, graph executor registers jmx mBean of running graph.</td>
<td>true</td>
</tr>
<tr>
<td>executor.debug_mode</td>
<td>If true, edges with enabled debug store data into files in debug directory. See property “graph.debug_path”</td>
<td>false</td>
</tr>
</tbody>
</table>

See “Clustering” section for more properties.
Chapter 17. Graph parameters

CloverETL Server passes set of parameters for each graph execution. Please keep in mind, that placeholders `${paramName}` are resolved only during loading of graph XML, so if you need placeholders resolving for each graph execution, graph cannot be pooled. However current parameter values are always accessible by inline java code like this:

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

Properties may be added or replaced like this:

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

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

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANDBOX_CODE</td>
<td>Code of sandbox which contains executed graph.</td>
</tr>
<tr>
<td>GRAPH_FILE</td>
<td>Path to the graph file, relative to sandbox root path. It is often referred as &quot;graphId&quot;.</td>
</tr>
<tr>
<td>SANDBOX_ROOT</td>
<td>Absolute path sandbox root.</td>
</tr>
<tr>
<td>RUN_ID</td>
<td>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 &quot;Launch Services&quot; for details.</td>
</tr>
</tbody>
</table>

Another sets of parameters according the type of execution

There are some more parameters in dependence of way, how the graph is executed.

executed from Web GUI

no more parameters

executed by Launch Service invocation

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

executed by HTTP API run graph operation invocation

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

executed by RunGraph component

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

no more parameters

executed by task "graph execution" by scheduler

Table 17.2. passed parameters

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_schedule_event_type</td>
<td>Type of schedule SCHEDULE_PERIODIC</td>
</tr>
<tr>
<td>event_schedule_last_event</td>
<td>Date/time of previous event</td>
</tr>
<tr>
<td>event_schedule_description</td>
<td>Schedule description, which is displayed in web GUI</td>
</tr>
<tr>
<td>event_username</td>
<td>User who &quot;owns&quot; the event. For schedule it is the user who created the schedule</td>
</tr>
<tr>
<td>event_schedule_id</td>
<td>ID of schedule which triggered the graph</td>
</tr>
</tbody>
</table>

executed by task "graph execution" by graph event listener

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

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

Table 17.3. passed parameters

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_run_sandbox</td>
<td>Sandbox with graph, which is source of the event</td>
</tr>
<tr>
<td>event_graph_event_type</td>
<td>GRAPH_STARTED</td>
</tr>
<tr>
<td>event_run_graph</td>
<td>graphId of the graph, which is source of the event</td>
</tr>
<tr>
<td>event_run_id</td>
<td>ID of the graph execution, which is source of the event.</td>
</tr>
<tr>
<td>event_timeout</td>
<td>Number of miliseconds which specifies interval of timeout. Makes sence only for &quot;timeout&quot; graph event.</td>
</tr>
<tr>
<td>event_run_result</td>
<td>Result (or current status) of the execution, which is source of the event.</td>
</tr>
<tr>
<td>event_username</td>
<td>User who &quot;owns&quot; the event. For graph events it is the user who created the graph event listener</td>
</tr>
</tbody>
</table>

executed by task "graph execution" by file event listener

Table 17.4. passed parameters

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_file_path</td>
<td>Path to file, which is source of the event. Does not contain file name. Does not end with file separator.</td>
</tr>
</tbody>
</table>
### How to add another graph parameters

**Additional "Graph Config Parameters"**

It is possible to add so-called additional parameters in Web GUI - section Sandboxes for the selected graph or for all graphs in the selected sandbox. See details in the section called “Graph config properties” (p. 21).

**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.

<table>
<thead>
<tr>
<th>key</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_file_name</td>
<td>Filename of the file which is source of the event.</td>
</tr>
<tr>
<td>event_file_event_type</td>
<td>SIZE</td>
</tr>
<tr>
<td>event_file_pattern</td>
<td>Pattern specified in file event listener</td>
</tr>
<tr>
<td>event_file_listener_id</td>
<td></td>
</tr>
<tr>
<td>event_username</td>
<td>User who &quot;owns&quot; the event. For file events it is the user who created the file event listener</td>
</tr>
</tbody>
</table>
Chapter 18. Recommendations for transformations developers

Add external libraries to app-server classpath

i.e. connections (JDBC/JMS) may require third party libraries. It is strongly recommended to add these libraries to app-server classpath.

CloverETL allows you to specify these libraries directly in graph definition so CloverETL may load these libraries dynamically, but external libraries may cause memory leak resulting with "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 server environment, all graphs are executed in the same VM instance. Attribute "same instance" of RunGraph component cannot be set to false.
Chapter 19. Logging

Main logs

CloverETL Server uses log4j library for logging. WAR file contains default log4j configuration.

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

"java.io.tmpdir" usually contains common system temp dir i.e. "/tmp". On tomcat, it is usually "[TOMCAT_HOME]/temp"

Default logging configuration may be overridden by system property "log4j.configuration", which should contain URL to log4j config file.

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

Since such configuration overrides 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>
```

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

client side:

```
```
(for more information consult CloverETL Designer User's Guide - chapter Integrating CloverETL Designer with CloverETL Server)

server side:

```
```

Graph run logs

Each graph run has its own log file, which is accessible i.e. in web GUI, section "executions history".

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

It is possible to specify different location for these logs by CloverETL property "graph.logs_path". This property does not have any influence over main server logs.
Chapter 20. Extensibility (Embedded OSGi framework)

Since 3.0

CloverETL Server includes embedded OSGi framework which allows implementation of "plugin" (OSGi bundle) which works as new API (or even GUI) of the server and it is independent of released clover.war.

Plugin possibilities

Basically the plugin may work as new server API similarly as Launch Services, HTTP API, WebServices API. It may be just simple JSP, HttpServlet or complex SOAP Web Services. So if the plugin contains some HTTP service, it is registered to listen on specified URL during the startup and incoming HTTP requests are "bridged" from the web container to the plugin. Plugin itself has access to the internal CloverETL Server interface called "ServerFacade". ServerFacade offers methods for execution graphs, obtaining of graph status and executions history, manipulation with scheduling, listeners, configuration and many more. So the API may be customized according to the needs of specific deployment.

Deploying an OSGi bundle

There are 2 CloverETL Server configuration properties related to the OSGi framework.

- plugins.path - Absolute path to the directory containing all your plugins (jar files).

- plugins.autostart - It is comma separated plugin names list. These plugins will be started during server startup. Theoretically OSGi framework can start the OSGi bundle on demand, however it is unreliable when the servlet bridge to the servlet container is used, so it is strongly recommended to name all your plugins.

So do deploy your plugin: set two config properties, copy plugin to the directory specified by "plugins.path" and restart the server.
Chapter 21. Extensibility CloverETL engine plugins

Since 3.1.2

CloverETL Server may use external engine plugins loaded from specified source. Source is specified by config property "engine.plugins.src"

See details about possibilities of CloverETL configuration in Chapter 16, Configuration (p. 81)

This property must be absolute path to the directory or zip file with additional CloverETL engine plugins. Both directory and zip must contain subdirectory for each plugin. These plugins are not a substitute for plugins packed in WAR. Changes in the directory or the ZIP file apply only when the server is restarted.

Each plugin has its own class-loader which uses parent-first strategy by default. 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 couple of ways how to ged rid of such conflicts:

- Remove your conflicting third-party libraries and use libraries on parent classloaders (web-app or app-server classloaders)

- Use different class-loading strategy for your plugin.
  - in the plugin descriptor plugin.xml, set attribute greedyClassLoader="true" in the element "plugin"
  - it means, that plugin classloader will use self-first strategy

- Set inverse class-loading strategy for selected java packages.
  - In the plugin descriptor plugin.xml, set attribute "excludedPackages" in the element "plugin".
  - It's comma separated list of package prefixes. E.g. like this: excludedPackages="some.java.package,some.another.package"
  - In 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.

Of course, 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 convinient debugging it is 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 "Logging" section for details about overriding server log4j configuration.
Chapter 22. Clustering

CloverETL Server only works in the cluster if the user's license allows it.

There are two common cluster features, high availability and scalability. Both of them are implemented by CloverETL Server on various levels. This section should clarify the basics of CloverETL Clustering.

High Availability

Since version 3.0, 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. This includes sandbox browsing, modification of services configuration (scheduling, launch services, listeners) and primarily graph executions. Any cluster node may accept incoming HTTP requests and process them itself or delegate it to another node.

Since all nodes are equal, almost all requests may be processed by any cluster node:

• All graph files, metadata files, etc. are located in shared sandboxes. Thus all nodes have access to them. A shared filesystem may be a SPOF, thus it is recommended to use a replicated filesystem instead.

• The database is shared by all cluster nodes. Again, a shared DB might be a SPOF, however it may be clustered as well.

But there is still a possibility, that a node cannot process a request by itself. In such cases, it completely and transparently delegates the request to a node which can process the request.

These are the requests which are limited to one (or more) 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 to a target node, however, this target node must be up and running.

• A graph is configured to use a partitioned or local sandbox. These graphs need nodes which have a physical access to the required sandboxes.

Thus an inaccessible partitioned or local sandbox may cause a failure from the request, however...

1. it is still possible to configure redundant sandboxes stored on other cluster nodes.

2. these types of sandboxes are used only for scalability on the data level (described below), which is a different approach to using a CloverETL cluster.

CloverETL itself implements a load balancer for executing graphs. So a graph 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 current load, which node will process the graph. All this is done transparently.

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

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.

Node Allocation

Node allocation is the specification of which cluster nodes will run the graph and which parts of the graph they will run. Allocation is basically specified by the partitioned sandboxes used in the graph phase. Each phase may have its own (just one) allocation. Basically, each partitioned sandbox has a list of locations. When some part of the graph runs in parallel, there is one worker for each partitioned sandbox location. See "Partitioned sandbox" in Partitioned and Local Sandboxes (p. 104) for details.

Allocation is specified in the graph either by:

- sandbox resources pointing to a partitioned sandbox, if workers read/write some partitioned data to/from their own location of this partitioned sandbox, or by

- the node attribute "node allocation", if workers do not read/write their partitioned data, however there must be an allocation specified.

If there is a conflict, execution fails and an error message appears containing the description of the conflict. A single conflict may be caused by using two different allocations in a single phase.

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 two special components to consider: ClusterPartitioner and ClusterGather. Both work similarly, but in the opposite way.

ClusterPartitioner works like a common partitioner, but node allocation is applied simultaneously behind the ClusterPartitioner component. All components preceding the ClusterPartitioner run on just one node (so called the primary worker - see below) whereas components behind the ClusterPartitioner run in parallel according to node allocation. Thus, these nodes work with just part of the data. There are more partitioning types: "round-robin" (default), "by record key", and "by load".
ClusterGather works in the opposite way. Components preceding the gather run in parallel while components behind the gather run on just one node (primary worker). The cluster gather component gathers records in the same way as SimpleGather and its attributes are the same. By default it does not sort input records in any way. It just gathers them in the order they come.

**Primary worker node** - some parts of the graph designed to run in parallel may run on a single node anyway, i.e. the part where the graph reads/writes data from/to a single resource. It may be the part preceding ClusterPartitioner or the part behind ClusterGatherer respectively. It also may be on all components in the phase which do not have node allocation specified at all. Each phase may have its own primary worker. All graph primary workers are chosen during graph execution primarily according to the local sandbox datasources used in the phases. Basically, the node which has direct(local) access to a sandbox datasource(s) used in the phase is selected as the primary worker. Of course, there may be multiple different local sandbox datasources, or even no local sandbox datasources used in the phase. In such cases, the server uses some minor parameters to choose the primary worker.

Both components may be combined in a single phase in any way, but there must be just one node allocation and just one primary worker in each single phase.

This example shows how data would be processed in 2 different node allocations, on 2 different primary workers.

- phase 1 starts
  - processing data on primary worker (nodeA)
  - cluster partitioner component
  - processing data in parallel (nodeA, nodeB, nodeC)
  - cluster gatherer component
  - processing data on primary worker (nodeA)
- phase 1 ends
- phase 2 starts
  - processing data on primary worker (nodeA)
  - cluster partitioner component
  - processing data in parallel (nodeB, nodeD)
- phase 2 ends
- phase 3 starts
  - processing data in parallel (nodeB, nodeD)
  - cluster gatherer component
  - processing data on primary worker (nodeD)
- phase 3 ends

Results are stored on a different node (nodeD) then the node that read (nodeA) and data is actually repartitioned (from nodeA, nodeB, nodeC to nodeB, nodeD).

**Partitioning/gathering data by external tools**

Partitioning data on the fly may in some cases be an unnecessary bottleneck. Splitting data using low-level tools can be much better for scalability. The optimal case being, that each running worker reads data from an independent
data source. Thus there does not have to be a ClusterPartitioner component in the first phase and the graph runs in parallel from the beginning.

- phase 1 starts
  - processing data in parallel (nodeA, nodeB, nodeC)
  - cluster gatherer component
  - processing data on primary worker (nodeA)
- phase 1 ends

Or the whole graph may run in parallel, however the results would be partitioned.

- phase 1 starts
  - processing data in parallel (nodeA, nodeB, nodeC)
- phase 1 ends

**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, metadata, connections, classes and input/output data for graphs which should support HA, as described above.

![Create new sandbox](image)

**Figure 22.1. Dialog form for creating new shared sandbox**

As you can see in the screenshot above, you cannot specify any root path on the filesystem. Shared sandboxes are stored in the directory specified by "cluster.shared_sandboxes_path". Each shared sandbox has its own subdirectory in it, which is named by sandbox ID.

**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 was a local file. See [Using a Sandbox Resource as a Component Data Source](p. 105) 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.

**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 graph/phase 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.

As you can see on the screenshot above, for a partitioned sandbox, you can specify one or more physical locations on different cluster nodes.

Do not use partitioned sandbox for common project data (graphs, metadata, connections, lookups, properties files, etc.). It would cause odd behavior. Use shared sandboxes instead.

**Using a Sandbox Resource as a Component Data Source**

A sandbox resource, whether it is a shared, local or partitioned sandbox (or ordinary sandbox on standalone server), is specified in the graph under the fileURL attributes as a so called sandbox URL like this:

```
sandbox://data/path/to/file/file.dat
```

where "data" is a code for sandbox and "path/to/file/file.dat" is the path to the resource from the sandbox root. URL is evaluated by CloverETL Server during graph 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 graph does not have to run on the node which has local access to the resource. There may be more sandbox resources used in the graph and each of them may be on a different node. In such cases, CloverETL Server would choose the node with the most local resources to minimalize remote streams.

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.

---

**Figure 22.2. Dialog form for creating new local sandbox**
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 as "license.file" for this property on Tomcat. Do not use clover_license.war. Tomcat loads web-apps in an unpredictable order and for the cluster, the license must be loaded before CloverETL Server itself.

Figure 22.3. List of nodes joined to the cluster

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 is only one phase and there are four cluster components in the graph (highlighted by red border). 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. This part of the graph performs data processing in parallel. This means that the components inside the dotted rectangle run on cluster nodes according to the "node allocation" of that part of the graph.

The rest of the graph runs just on one node called "primary worker".

**Specification of "node allocation"**

Since there is only one phase, the whole graph has just one primary worker and only one node allocation.

- node allocation is applied for groups of components running in parallel (demarcated by the four cluster components)
- the outer part of the graph run on a single node - primary worker.

The primary worker 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 primary worker in the graph.

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 "allocation" attribute. Since all components in this part must have the same allocation, it is sufficient to set it only for one component.

Again, "dataPartitioned" in the following dialog is the sandbox ID.
Let's investigate our sandboxes. This project requires 3 sandboxes: "data", "dataPartitioned" and "PhoneChargesDistributed".

- **data**
  - contains input and output data
  - local sandbox (yellow folder), so it has only one physical location
  - accessible only on node "i-4ce9733b" in the specified path

- **dataPartitioned**
  - partitioned sandbox (red folder), so it has a list of physical locations on different nodes
  - does not contain any data and since the graph does not read or write to this sandbox, it is used only for the definition of "nodes allocation"
  - on the following figure, allocation is configured for two cluster nodes

- **PhoneChargesDistributed**
  - common sandbox containing the graph file, metadata, and connections
  - shared sandbox (blue folder), so all cluster nodes have access to the same files
If the graph was executed with the sandbox configuration of the previous figure, the node allocation would be:

- components which run only on primary worker, 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" 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:
Chapter 22. Clustering

Figure 22.4. 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:

\[
speedupFactor = \frac{\text{avgRuntime}(1 \text{ node})}{\text{avgRuntime}(x \text{ nodes})}
\]

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 22.5. Speedup factor

Table of measured runtimes:

<table>
<thead>
<tr>
<th>nodes</th>
<th>runtime 1 [s]</th>
<th>runtime 2 [s]</th>
<th>runtime 3 [s]</th>
<th>average runtime [s]</th>
<th>speedup factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>861</td>
<td>861</td>
<td>861</td>
<td>861</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>467</td>
<td>465</td>
<td>466</td>
<td>466</td>
<td>1.85</td>
</tr>
<tr>
<td>3</td>
<td>317</td>
<td>319</td>
<td>314</td>
<td>316.67</td>
<td>2.72</td>
</tr>
<tr>
<td>4</td>
<td>236</td>
<td>233</td>
<td>233</td>
<td>234</td>
<td>3.68</td>
</tr>
<tr>
<td>5</td>
<td>208</td>
<td>204</td>
<td>204</td>
<td>205.33</td>
<td>4.19</td>
</tr>
<tr>
<td>6</td>
<td>181</td>
<td>182</td>
<td>182</td>
<td>181.67</td>
<td>4.74</td>
</tr>
<tr>
<td>7</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>168</td>
<td>5.13</td>
</tr>
<tr>
<td>8</td>
<td>172</td>
<td>159</td>
<td>162</td>
<td>164.33</td>
<td>5.24</td>
</tr>
</tbody>
</table>
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 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. 111)
- Optional properties (p. 112)
- Example of 2 node cluster configuration (p. 112)
- Load balancing properties (p. 113)

**Mandatory properties**

*Table 22.1. Mandatory properties - these properties must be properly set on each node of the cluster*

<table>
<thead>
<tr>
<th>property</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.enabled</td>
<td>boolean</td>
<td>false</td>
<td>switch whether server is connected to the cluster or not</td>
</tr>
<tr>
<td>cluster.node.id</td>
<td>String</td>
<td>node01</td>
<td>each cluster node must have unique ID</td>
</tr>
<tr>
<td>cluster.shared_sandboxes_path</td>
<td>String, path</td>
<td></td>
<td>Path, where all shared sandboxes are stored on this node. If cluster is enabled, all sandboxes are shared, thus &quot;rootPath&quot; attribute of the sandbox is ignored. Path to the root directory of the sandbox is constructed like this: [shared_sandboxes_path]/[sandboxID]</td>
</tr>
<tr>
<td>cluster.jgroups.bind_address</td>
<td>String, IP address</td>
<td>127.0.0.1</td>
<td>IP address of ethernet interface, which is used for communication with another cluster nodes. Necessary for inter-node messaging.</td>
</tr>
<tr>
<td>cluster.jgroups.start_port</td>
<td>int, port</td>
<td>7800</td>
<td>Port where jGroups server listens for inter-node messages.</td>
</tr>
<tr>
<td>cluster.jgroups.tcping.initial_hosts</td>
<td>String, in format: &quot;IPaddress1[port1],IPaddress2[port2]&quot;</td>
<td>127.0.0.1[7800]</td>
<td>List of IP addresses(with ports) where we expect running and listening nodes. It is related to another nodes &quot;bind_address&quot; and &quot;start_port&quot; 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.</td>
</tr>
<tr>
<td>cluster.http.url</td>
<td>String, URL</td>
<td><a href="http://localhost:8080/clover">http://localhost:8080/clover</a></td>
<td>URL to the root of web application of configured node. Necessary for inter-node cooperation. This value will be sent to all other nodes in the cluster to let them know how to connect to this node.</td>
</tr>
</tbody>
</table>
## Optional properties

*Table 22.2. Optional properties - these properties aren’t vital for cluster configuration - default values are sufficient*

<table>
<thead>
<tr>
<th>property</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.node.sendinfo.interval</td>
<td>int</td>
<td>5000</td>
<td>time interval in ms; each node sends info about itself to another nodes; this interval specified how often the info is sent</td>
</tr>
<tr>
<td>cluster.node.remove.interval</td>
<td>int</td>
<td>15000</td>
<td>time interval in ms; if no node info comes in this interval, node is considered as lost and it is removed from the cluster</td>
</tr>
<tr>
<td>cluster.max_allowed_time_shift_between_nodes</td>
<td>int</td>
<td>2000</td>
<td>Max allowed time shift between nodes. If time shift exceeds this, node will be selected as invalid.</td>
</tr>
<tr>
<td>cluster.group.name</td>
<td>String</td>
<td>cloverCluster</td>
<td>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.</td>
</tr>
<tr>
<td>cluster.max_allowed_time_shift_between_nodes</td>
<td>int</td>
<td>2000</td>
<td>How many milliseconds is maximum allowed time shift between nodes in the cluster. 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.</td>
</tr>
</tbody>
</table>

## Example of 2 node cluster configuration

This section contain example of CloverETL cluster nodes configuration. In addition it is necesssary to configure:

- sharing or replication of directory /home/clover/nfs_shared/sandboxes
- connection to the same database from both nodes
- HTTP load balancer

Configuration of node on 192.168.1.131

```java
dialect=org.hibernate.dialect.MySQLDialect
datasource.type=JNDI
datasource.jndiName=java:comp/env/jdbc/clover_server
cluster.enabled=true
cluster.node.id=node01
cluster.shared_sandboxes_path=/home/clover/nfs_shared/sandboxes
license.file=/home/clover/license/license.dat
```
Chapter 22. Clustering

Configuration of node on 192.168.1.13

```java
cluster.group.name=cloverCluster
cluster.jgroups.bind_address=192.168.1.13
cluster.jgroups.start_port=7800
cluster.jgroups.tcpping.initial_hosts=192.168.1.13[7800]
```

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 22.3. Load balancing properties

<table>
<thead>
<tr>
<th>property</th>
<th>type</th>
<th>default</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster.lb.balance.running_graphs</td>
<td>float</td>
<td>3</td>
<td>Specify importance of running graphs for load balancing.</td>
</tr>
<tr>
<td>cluster.lb.balance.memused</td>
<td>float</td>
<td>0.5</td>
<td>Specify importance of used memory for load balancing.</td>
</tr>
<tr>
<td>cluster.lb.balance.cpus</td>
<td>float</td>
<td>1.5</td>
<td>Specify importance of number of CPUs for load balancing.</td>
</tr>
<tr>
<td>cluster.lb.balance.master_bonus</td>
<td>float</td>
<td>1</td>
<td>Specify importance of the fact, that the node is master. Usually it does not affect anything, thus value 1 says to load balancer: &quot;consider master node the same as any other node&quot;</td>
</tr>
<tr>
<td>cluster.lb.balance.this_node</td>
<td>float</td>
<td>2</td>
<td>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.</td>
</tr>
</tbody>
</table>
List of Figures

2.1. Clover Server as the only running application on IBM Websphere ........................................ 12
2.3. Websphere7 configuration ........................................................................................................ 68
3.1. Sandboxes Section in CloverETL Server Web GUI ................................................................. 15
3.2. Websphere configuration .......................................................................................................... 67
3.3. Sandboxes Section in CloverETL Server Web GUI ................................................................. 16
3.4. Sandbox Permissions in CloverETL Server Web GUI ........................................................... 17
3.5. Web GUI - section "Schedules" ................................................................................................. 22
3.6. Web GUI - upload ZIP to sandbox ......................................................................................... 19
3.7. Web GUI - upload ZIP results ............................................................................................... 19
3.8. Web GUI - download file in ZIP ............................................................................................ 20
3.9. Graph config properties ......................................................................................................... 23
4.1. Web GUI - section "Users" ....................................................................................................... 27
4.2. Web GUI - edit user ................................................................................................................. 27
4.3. Web GUI - change password .................................................................................................. 28
4.4. Web GUI - groups assignment ............................................................................................... 28
4.5. Web GUI - section "Groups" .................................................................................................. 29
4.6. Web GUI - groups assignment ............................................................................................... 29
4.7. Tree of permissions ................................................................................................................. 30
5.1. Web GUI - section "Scheduling" - create new ......................................................................... 31
5.2. Web GUI - onetime schedule form ......................................................................................... 32
5.3. Web GUI - schedule form - calendar ...................................................................................... 32
5.4. Web GUI - periodical schedule form ....................................................................................... 33
5.5. Cron periodical schedule form ............................................................................................... 34
5.6. Web GUI - Graph execution task .......................................................................................... 35
5.7. Web GUI - "Kill graph" ........................................................................................................... 36
5.8. Web GUI - shell command ..................................................................................................... 36
5.9. Web GUI - archive records ..................................................................................................... 39
6.1. Web GUI - graph timeout event ............................................................................................. 41
6.2. Web GUI - send email ............................................................................................................. 43
6.3. Web GUI - Task JMS message editor ..................................................................................... 45
6.4. Event source graph isn't specified, thus listener works for all graphs in specified sandbox .... 46
6.5. Web GUI - email notification about graph failure ................................................................. 46
6.6. Web GUI - email notification about graph success ............................................................... 47
6.7. Web GUI - backup of data processed by graph ...................................................................... 47
9.1. Web GUI - "Manual task execution" section ............................................................................ 53
10.1. Web GUI - "File event listeners" section .............................................................................. 54
13.1. Glassfish JMX connector ....................................................................................................... 66
13.2. Websphere configuration ....................................................................................................... 67
13.3. Websphere7 configuration ...................................................................................................... 68
15.1. Launch Service Overview ..................................................................................................... 74
15.2. Launch Service section ......................................................................................................... 76
15.3. The Basic Info tab ............................................................................................................... 76
15.4. Edit Configuration tab ......................................................................................................... 77
15.5. Edit Parameters tab ............................................................................................................. 77
15.6. Edit Parameters tab ............................................................................................................. 78
15.7. Edit Parameters tab ............................................................................................................. 78
22.1. Dialog form for creating new shared sandbox .................................................................. 104
22.2. Dialog form for creating new local sandbox ...................................................................... 105
22.3. List of nodes joined to the cluster ....................................................................................... 106
22.4. Cluster Scalability ............................................................................................................... 110
22.5. Speedup factor .................................................................................................................... 110