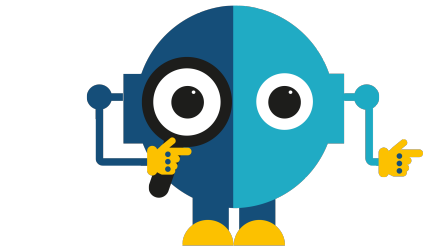

D.O.T

Configuration Guide

DOT Extract Server

Version 24.0

Publication Date: January, 2024

Prepared by the DOT Software Documentation Team

North America & LATAM
1 N. State St, 15th Floor
Chicago, IL
USA
1-603-371-9074
1-603-371-3256 (support calls only)
sales-us@arcadsoftware.com

EMEA (HQ)
55 Rue Adrastée – Parc Altaïs
74650 Chavanod/Annecy
France
+33 450 578 396
sales-eu@arcadsoftware.com

Asia Pacific
5 Shenton Way #22-04
UIC Building
Singapore 068808
sales-asia@arcadsoftware.com

Preface

Document purpose

This document is intended to guide you through the configuration of Extract Server.

Intended Audience

This document is intended for administrators and/or the person in charge of high level configuration of Extract Server.

Related Documentation

Related Documentation
Release Notes

Table 1: Related Documentation

Unless stated otherwise, all content is valid for the most current version of DOT Extract Server listed as well as every subsequent version.

Product Version	Document Version	Publication Date	Update Record
24.0	1.7	January, 2024	No functional changes
23.2	1.6	July, 2023	General documentation update
23.1	1.5	April, 2023	General documentation update
23.0	1.4	January, 2023	Added the configuration tools

Table 2: DOT Extract Server Configuration Guide Publication Record

Contact DOT Software

Headquartered in France at the foot of the Alps, DOT Software offers global services and has offices and partners all over the world. ARCAD partners with leading-edge companies throughout the world to offer full services, close to home.

Visit our website to Contact Us and find out more about our company and partners, or to request a demo.

The ARCAD Customer Portal is intended for current and potential customers that have full or trial versions of ARCAD software. If you already use or are interested in using an ARCAD product, the portal lets you view all of your current licenses and generate your own temporary license keys for most ARCAD products. It grants you access to the ARCAD product knowledge base (new releases, release notes and current documentation).

Do you have a request for change or have you encountered a bug? Log into the ARCAD Helpdesk and create a ticket.

DOT Software guarantees consultant support 24 hours a day, 5 days a week (24/5) to registered members. Calls received are redirected, according to the hour, to put you in contact with a support team in or near your timezone.

<table>
<tr><th>Country</th><th>Address</th><th>Account Contact</th><th>Support Contact</th></tr>
<tr><td>France</td><td>ARCAD Software (HQ)
55 Rue Adrastée
74650 Chavanod

ARCAD Software
17 chemin de la plaine
07200, Saint-Didier-sous-Aubenas</td><td rowspan="2">+33 4 50 57 83 96
sales-eu@arcadsoftware.com</td><td rowspan="6">Worldwide 24/7:
+1 603 371 3256

France only:
+33 450 57 28 00

support@arcadsoftware.com

ARCAD Helpdesk</td></tr>
<tr><td>Germany</td><td>ARCAD Software Deutschland GmbH
c/o Pramex International GmbH
Savignystr. 43, 60325 Frankfurt am Main</td></tr>
<tr><td>China</td><td>ARCAD Software
#2035, Yuehai Plaza,
180 Wanbo 2nd Road,
Nancun, Panyu District, Canton</td><td rowspan="3">+86 (020)22324643
+86 (020)22324649
sales-asia@arcadsoftware.com</td></tr>
<tr><td>India</td><td>ARCAD Software
D-280/281/282, Vibhuti Khand
Gomti Nagar, Lucknow</td></tr>
<tr><td>Singapore</td><td>ARCAD Software
5 Shenton Way #22-04
UIC Building
Singapore 068808</td></tr>
<tr><td>USA</td><td>ARCAD Software
1 N. State St, 15th Floor
Chicago, IL</td><td>+1 (603) 371-9074
+1 (603)-371-3256 (support calls only)
sales-us@arcadsoftware.com</td></tr>
</table>

Table 3: Contact DOT Software

Contents

1 Overview

This document is intended to guide you through the configuration of DOT Extract Server. The main configuration is done through a set of configuration files (with extension**.cfg** or **.ini**) stored in the configuration folder under the application's home directory. Every path given here is relative to the application home directory.

1.1 First run check list

The following steps must be checked before running the application for the first time. Incorrect configuration may compromise the stability of the application or require a new installation.

Make sure that:

1. The *./configuration* folder contains the **config.ini** file, the **osgi.cm.ini** file and all the ***.cfg** files used for specific configuration parts. Note that the folder containing the *.cfg files is designated in the config.ini file by the **osgi.cm.storefile** parameter.
2. The **com.arcadsoftware.masterkey** property or the **com.arcadsoftware.masterkey.fog** property in the config.ini file are set to a value complex enough. These parameters are used to generate all encrypted materials.

> ***Warning!***
> If these parameters are not set before the first run of the application, a random value will be used and stored into the config.ini file.
>
> Their value must not be changed after the first run of the application in any case. If changed, important information and encrypted data will be lost.

3. In the database parameters, the data sources required by the application are predefined to connect to an H2 database. Make sure that the H2 database file itself exists and the default password of the JDBC connection must be been changed to a more complex one. If you plan to use a PostgreSQL database, you must install it before starting the server of the application.
4. In the HTTP(S) server configuration, the HTTP and/or HTTPS port numbers are set. These TCP port numbers must be free and authorized by any firewall control. If a HTTPS server is used, make sure that the keystore is set and points to an existing file containing the correct certificate associated to your company policy.
5. The log is configured so that a log file is generated. For a first run, the log level can be set to the **info** level, but in a production environment, the log level should be set to the **warn** or **error** levels.

These check points ensure that the application starts correctly. However, the application may require some specific additional configuration steps. Some can be done before the first run, like installing a license key.

1.2 General documentation remarks

The parameters that may compromise the security of the application are indicated in **Warning** boxes. These options should not be activated in production, except for diagnostic purposes and for a limited time.

For each parameter, when necessary, the type of parameter is indicated. The most common parameter types are strings, integers and true/false for Boolean values. The expression of these values may change depending on the format of the configuration file.

> **Note**
> When a default value is specified in the description of a parameter, this parameter is an optional one.

> **Important!**
> Setting a parameter to an empty value does not set this parameter to the default value. To do so, the parameter must not be present in the configuration files.

2 Configuration tools

In the application installation subfolder *./tools* comes a set of command line programs which may facilitate the configuration of the application or diagnose the state of the installation.

These programs must be used only when the application program is turned off. Moreover, these commands must be run with the same user and privileges on the operating system as the ones of the application to manage.

All the following commands accept the arguments:

`-v or -version`
: This argument disables the command action and prints the command version number. It is used to check that the command currently installed matches the following documentation.

`-h or -help`
: This argument disables the action of the command and prints the command help message similar to this documentation.

`-homedir <path>`
: This parameter defines the home directory of the application to manage. This parameter is automatically defined in the shell scripts (.sh and .bat). If required, you have to edit this script to change its value.

`-debug`
: This argument prints detailed information of the command execution. It is useful to trace the execution and detect where a problem occurs, if something goes wrong. It is not recommended to use it every time, because the printed message may obfuscate the important messages.

> ⚠ **Important!**
> These commands must be executed with the same user that runs the application server. On Windows, you have to execute it as Administrator, and on Linux you have to use the "sudo -u" command or any equivalent command.

Exit codes

The following exit codes may occur if a problem does not allow the completion of the command:

0	The command execution is a success
1-9	These exit codes are relative to the Java Virtual Machine. You should check the JVM installation.
32	An internal or unexpected error occurred during the command execution. The console message may include details relative to this problem. If not, try to relaunch the command with the **-debug** option.
33	The application home directory is not correct, or is not specified. Check the value of the `-homedir` option.

34	The config.ini file is corrupted, or not found. This file is required to identify the application parameters.
35	The application configuration is corrupted or not correctly declared into the config.ini file.
40	A required parameter is missing. Relaunch the command with the option `-help` to get the list of required parameters.
41	A parameter value is incorrect, of a set of incompatible parameters is used. Relaunch the command with the option `-help` to get more information about the command parameters.
50	The acceded database is corrupted or unavailable. Check the database connection parameters or try to restore a backup of it.
51	There is a problem with the file system. This may be a missing access right or a full disk space.
60	The operation has been canceled.
61	The test performed by the command has failed.

Other exit codes may be returned by the Java command or the operating system.

dbmigration

Command line:

Linux/Unix: `./dbmigration.sh`

Windows: `./dbmigration.bat`

This command allows the migration of the application database from H2 Database to PostgreSQL. This migration is a one-way operation and requires some specific steps before the execution, such as the installation and the configuration of the PostgreSQL database.

For more details about this command, refer to the Database installation and upgrade chapter.

This command uses the current configuration of the application to access the H2 Database. Depending on your installation, this may require starting the H2 Database server if it is run remotely. The connection parameter to the new PostgreSQL database can be passed through the command parameter or is prompted.

Parameters:

`<url>`

The first parameter of the command can be the JDBC URL used to connect to the target PostgreSQL database. This URL must have the following pattern: *jdbc:postgresql://<host>[:<port>]/<database>* where host is the PostgreSQL server hostname, port is the TCP port number, if different from the default one (i.e. 5432), and database is the PostgreSQL database name. Other options may be added to this URL. For more information about these options, refer to the official PostgreSQL JDBC documentation.

`-l <user>, -login <user>`

The user login used to connect to the PostgreSQL database. Note that this user must be associated to a default schema where the application database is created.

`-p <password>, -password <password>`

The corresponding user password.

-nb, -nobackup

If used, this optional parameter disables the backup of the source database before processing to the migration.

-ds <datasource id>, -datasource <datasource id>

Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters are used for any of them.

-ds.login <user id>, -datasource.login <user id>

Replaces the default user login of the data sources by the specified one. This may be required if the original user has limited rights on the database. It is recommended to use this parameter with the `-ds` one, if not, it is applied to all data sources.

-ds.pwd <password>, -datasource.password <password>

Parameter used with the `-ds.login` parameter. It defines the new user password. If this parameter is not used, the password is prompted.

dbpwd

Command line:

Linux/Unix: `./dbpwd.sh`

Windows: `./dbpwd.bat`

This command changes the password of the user connected to the application database. It changes the password in the Database (H2 Database or PostgreSQL) and in the configuration of the application server.

Parameters:

-pwd <password>, -password <password>

Defines the new password to be used.

-gen, -generate

If used, this parameter generates a random password for the data sources connection, ignoring any provided password and disabling any prompt.

-ds <datasource id>, -datasource <datasource id>

Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters using its parameters are used for any of them.

> ⚠ **Important!**
> It is highly recommended to change the password of the embedded database delivered with the application, before the first start of the application server, to ensure that no one can access to your data.

> ⚠ You can use the -gen option to ensure that the new password, do not forget to note this password if you want to have a an external access to your data. By the way, you will still able to change it again later.

dbupdate

Command line:

Linux/Unix: `./dbupdate.sh`

Windows: `./dbupdate.bat`

This command performs the database update during the installation of a new version of the application. It also generates a backup of the database, only for H2 Databases, before the update, allowing the restoration of the database state if something goes wrong.

For more details about this command, refer to the Database installation and upgrade chapter.

Parameters:

`-i, -install`
: This option forces the database creation if the current database target is an empty database.

`-np, -noprompt`
: This option disables the user prompt, asking for a manual backup of a non-H2 Database.

`-p <password>, -password <password>`
: This option defines the password to use for the H2 Database backup file. If this parameter is not set, the password used is the one used to connect to the database. If a blank password is given, then no password is used to encrypt the backup file, which is not recommended.

`-ncdb, -nocleandb`
: If used, this option disables the "clean up" process of the H2 database. Note that this process may restore the performance of the H2 database and test the validity of the backup archive, but it may also corrupt the database if an error occurs.

`-ds <datasource id>, -datasource <datasource id>`
: Defines the application data source to access, through its identifier. It may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters are used for any of them.

`-ds.login <user id>, -datasource.login <user id>`
: Replaces the default user login of the data sources by the specified one. This may be required if the original user has limited rights on the database. It is recommended to use this parameter with the `-ds` one, if not, it is applied to all data sources.

`-ds.pwd <password>, -datasource.password <password>`
: Parameter used with the `-ds.login` parameter. It defines the new user password. If this parameter is not used, the password is prompted.

encrypt

Command line:

Linux/Unix: `./encrypt.sh`

Windows: `./encrypt.bat`

This command allows securing the application installation, by the generation of a strong master key, and/or the encryption of a password in the configuration files.

Parameters:

`-gmk, -genmasterkey`
: If it is not already generated, this parameter generates a random master key used to encrypt all other secured data.

`-k [<pid>:]<property>, -key [<pid>:]<property>`
: If used, this parameter must match a property name of a configuration section (pid), the one in which you wish to change the password value.

`-p <new password>, -password <new password>`
: Specifies the password to encrypt. If this parameter is not defined, the password is prompted.

`-f, -fog`
: Uses the Fog encryption algorithm instead of the stronger one. Fog is not a secure algorithm but it is faster to decrypt and transferable to other locations.

If the key parameter is omitted, the encrypted password is printed on the console. If used, this parameter must define the property name and the "PID" of the configuration of this property. This pid corresponds to the name of the .cfg file containing this property. Note that this name differs from the section name used in the .ini file. Refer to the documentation to identify this name.

For instance, to define and encrypt the password used to import a user from IBM I in the *REST Connection AS400*, use the following command call:

```
./encrypt.sh -key com.arcadsoftware.server.restful.connection.as400:importPassword
```

h2backup

Command line:

Linux/Unix: `./h2backup.sh`

Windows: `./h2backup.bat`

This command generates a backup of the H2 database. By default, the backup file is named: *./database/backup/<data source ID>_<time stamp>.zip* and it is protected by a password that corresponds to the one used to connect to this database.

Parameters:

-p <password>, -password <password>
Specifies the password used to encrypt the backup file. By default, the H2 database connection password is used.

-f <file path>, -file <file path>
Defines a specific destination backup file path.

-ds <datasource id>, -datasource <datasource id>
Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters are used for any of them.

h2restore

Command line:

Linux/Unix: `./h2restore.sh`

Windows: `./h2restore.bat`

This command allows the restoration of an H2 database backup previously generated by the h2backup or dbmigration commands. This command destroys any existing H2 database before restoring the backup file.

Parameters:

-f <file path>, -file <file path>
Defines the backup file path. By default, the path is prompted.

-p <password>, -password <password>
Specifies the password used to encrypt the backup file. By default, the H2 database connection password is used.

-ds <datasource id>, -datasource <datasource id>
Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters are used for any of them.

masterkey

Command line:

Linux/Unix: `./masterkey.sh`

Windows: `./masterkey.bat`

This command generates a master key. This security material is used to encrypt all other passwords required by the application. The generated key is a random string or uses the legacy algorithm to generate a non-trivial but repeatable string. As the random string generates a stronger key, it is recommended to use the legacy algorithm (fog) only when the restoration of a previous installation is required.

Parameters:

-d, -default
: If this parameter is set, this command generates a key with the application default process.

-jh <path>, -javahome <path>
: This parameter is combined with the -default parameter. It replaces the current java home directory with the given one.

-hn <dns>, -hostname <dns>
: This parameter is combined with the `-default` parameter. It replaces the current hostname with the given one.

-on <name>, -osname <name>
: This parameter is combined with the `-default` parameter. It replaces the current OS name with the given one.

Refer to the Security and cryptography chapter for more details about the usage of the master key.

selfcerts

Command line:

Linux/Unix: `./selfcerts.sh`

Windows: `./selfcerts.bat`

This command generates self-signed certificates for the HTTPS server of the application. As this certificate is not validated by a certification authority (CA), you have to manually install it on the client workstations.

Parameters:

-ks <path>, -keystore <path>
: Defines the path to the keystore. If this parameter is not defined, the default value *./security/keystore.p12* is used.

-kst <type>, -keystoretype <type>
: Defines the type of keystore used. By default, the type is *PKCS12*.

-ksp <password>, -keystorepwd <password>
: Defines the password used for the keystore.

-rdp, -randompwd
: If a password is not defined, the password is randomly generated. Note that this option is only applicable with a new keystore.

-alias <alias>, -keyalias <alias>
: Defines the key alias in the keystore. By default, the value is *serverkey*. Note that this alias must be unique in the keystore.

-keypwd <password>, -keypassword <password>
: Defines the password of the key in the keystore.

-size <int>, -key.size <int>

Defines the private key size. By default, the value is *2048*.

`-key <name>, -keyalgorithm <name>`

Defines the private key algorithm. By default, the value is *RSA*, and accepted values are defined by the current version of Java used to run the application.

`-dns <dns>, -domainname <dns>`

This parameter corresponds to the domain name of this server, as the client has access to it. The default value is the actual DNS name of this host.

`-issuer <name>`

Defines the certificate Issuer name included in the server certificate.

`-org <name>, -organization <name>`

Defines the organization name included in the server certificate.

`-ou <name>, -organizationunit <name>`

Defines the organization unit name included in the server certificate.

`-loc <locality>, -locality <locality>`

Defines the locality name included in the server certificate.

`-c <code>, -country <code>`

Defines the country code name included in the server certificate. This code must comply with the country name standard (ISO 3166).

`-st <state>, -state <state>`

Defines the sate code name included in the server certificate.

`-em <address>, -email <address>`

Defines the contact email address included in the server certificate.

`-sign <name>, -signaturealgorithm <name>`

Defines the name of the algorithm used to generate the signature of the certificate. The default value is *SHA-256* with RSA encryption.

`-l <limit>, -limit <limit>`

Defines the certificate limitation of validity, this value is defined in days, or with a w, m or y suffix, respectively in weeks, months, or years. By default, the value is set to *10y*.

`-nct, -noclientstore`

Disables the generation of a truststore for the usage of the clients of this server.

`-ts <path>, -truststore <path>`

Defines the path to the client truststore. If not defined, the default value *./security/clienttruststore.p12* is used.

`-tst <type>, -truststoretype <type>`

Defines the type of client truststore used. By default, the type is *PKCS12*.

-tsp <password>, -truststorepwd <password>

Defines the password used for the client truststore. If this parameter is not defined and the `-randompwd` parameter is used, then the password is not prompted, it is randomly generated and printed on the screen.

setloginpwd

Command line:

Linux/Unix: `./setloginpwd.sh`

Windows: `./setloginpwd.bat`

This command allows you to change a user password stored in the application database. Any password set for a local login to connect to the application can be reset with this command, but the main usage of this parameter is to change the default user (i.e. the "Administrator") and the password included in a fresh installation of the application if there is any. To do so, no parameters are required, and the new password of the administrator is prompted.

Parameters:

-p <password>, -password <password>

Defines the new password to update in the database. This parameter is optional. If it is not specified, it is prompted in the console.

-id <integer>

Defines the internal user ID to update, as defined in the database. This parameter is optional. The default value is 1 (one), and the ID of the default user is included in the database if there is any.

-l <user login>, -login <user login>

The login of the user to update in the database. This parameter is optional. If it is not specified, the value of the `-id` parameter is used.

-ds <datasource id>, -datasource <datasource id>

Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or any data source defined by the application is applied to the command, and the same parameters will be used for any of them.

testds

Command line:

Linux/Unix: `./testds.sh`

Windows: `./testds.bat`

This command tests the configuration of the data source. It tests the connection to the database and the create, read, update and delete operations on the data.

Refer to the configuration of Data Sources for details.

Parameters:

-ds <datasource id>, -datasource <datasource id>

Defines the application data source to access, through its identifier. This may be required if the application defines multiple data sources, which is really uncommon. By default, the unique, or all data source defined by the application is applied to the command, and the same parameters are used for any of them.

testhttp

Command line:

Linux/Unix: `./testhttp.sh`

Windows: `./testhttp.bat`

This command tests the configuration of the HTTP and HTTPS servers. It checks that the TCP-IP port numbers are correct and free to use. It also checks the TLS configuration, the private key, the passwords, and the security parameters related to these servers.

Refer to the configuration of the HTTP Server REST for details.

testldap

Command line:

Linux/Unix: `./testldap.sh`

Windows: `./testldap.bat`

This command tests the configuration of the LDAP connection that is used to authenticate the application user. To do so, this command tries to bind a login/password to the LDAP server, just like the application user does. Then it tries to perform a research into the LDAP tree and returns the number of potential objects corresponding to users.

Refer to the configuration of the REST Connection LDAP for details.

Parameters:

-l <user login>, -login <user login>

The user login used to test the LDAP binding. This parameter is optional. If it is not specified, it is prompted during the test.

-p <pass phrase>, -password <pass phrase>

The user password used to test the LDAP binding. This parameter is optional. If it is not specified, it is prompted during the test.

-sp <ldap search pattern>, -search <ldap search pattern>

Defines the search pattern to be used during the test. If this parameter is not defined, a global selection is used (search pattern corresponding to "*").

Uninstall

Command line:

Linux/Unix: `./uninstall.sh`

Windows: `./uninstall.bat`

This command removes the *tools* folder. Once the configuration tools are no longer required, you can safely remove them by deleting this folder or running this command.

Parameters:

This command does not have any specific parameters.

Updatecfgini

Command line:

Linux/Unix: `./updatecfgini.sh`

Windows: `./updatecfgini.bat`

This command is used during the application update. It allows to inject the system properties declared into the previous config.ini file into the new one. By default only the critical properties are copied, and only if they do not already exist into the new config.ini file.

This command is indented to be called during the application update, there is no need to use it in a common installation.

Arguments:

`-a, -all`
: Imports all properties from the old config.ini file except the bundles path related ones.

`-o, -override`
: Overrides the existing properties in the new config.ini by the old values.

`-ci <path>, -configini <path>`
: Defines the path of the previous config.ini file. The default value is *./configuration/config.bak*.

3 Data Sources

Configuration located in the *com.arcadsoftware.database.sql.cfg* file or in the section *[Data Sources]* of the *osgi.cm.ini* file.

In this section, manage the parameters used to connect to the database(s) of the application.

In this section, define the data sources required by the application. The names of the data sources are specific to the application therefore they must be correctly defined, they must point to existing databases and their names cannot be changed. If you decide to migrate to a different database, you need to change these parameters.

The name used in the keys of the following parameters, represented by the placeholder **dsname**, needs to be replaced by the actual data source name defined by the application.

dsname.dbid

Parameter	Type	Default value
dsname.dbid	String	*dsname*

This parameter identify the data source. The value of the parameter must be equals to *dsname* prefix.

> **Note**
> This deprecated parameter is no longer required.

dsname.type

Parameter	Type	Default value
dsname.type	String	

Sets the type of database and JDBC driver used. Supported values are:

- **h2** or **h2e**: is used for a single connection to a H2 database.
- **h2p**: is used for a pooled connection to a H2 database.
- **postgresql**: is used for connections to PostgreSQL databases.

This parameter is optional, when the connection type can be deducted from the JDBC URL. Refer to the configuration of *dsname*.url.

dsname.url

Parameter	Type	Default value
dsname.url	URL	

Sets the JDBC URL used to connect to the database. The value depends on the JDBC driver used:

- If the data source type is one of the H2 types, this parameter is optional and the database URL is then **jdbc:h2:./database/*dsname***. If set, this URL can be just the path to the database location, the database is then used as an embedded one. In other cases the URL must start with **jdbc:h2:**.

- For an PostgreSQL database, the URL must follow the pattern: **jdbc:postgresql://<host name>[:<port>]/<database name>**.

dsname.login

Parameter	Type	Default value
dsname.login	String	

Sets the user login used to connect to the database.

If the data source type is one of the H2 types, this parameter is optional. The default value is **sa**.

If the data source type is PostgreSQL, it is required to define this parameter.

dsname.password

Parameter	Type	Default value
dsname.password	Encrypted string	

Sets the user password used to connect to the database. This value can be encrypted.

dsname.timeout

Parameter	Type	Default value
dsname.timeout	Integer	30,000

Sets, in milliseconds, the duration of the connection timeout.

dsname.poolmin

Parameter	Type	Default value
dsname.poolmin	Integer	0

Sets, in case of a pooled set of connections, the minimum number of concurrent connections opened.

dsname.poolmax

Parameter	Type	Default value
dsname.poolmax	Integer	0

Sets, in case of a pooled set of connections, the maximum number of concurrent connections opened.

dsname.dialect

Parameter	Type	Default value
dsname.dialect	String	

Sets the SQL dialect used by this data source. The dialect defines the SQL order that the data mapping implementation uses to communicate with the database. There is a table that automatically sets the value of this parameter, so it does not need to be set. The value is case-sensitive and the accepted values are the following: **H2**, **Firebird**, **DB2**, **DB2400**, **Oracle**, **MsSQL**, **MySQL**, **Derby**, **PostgreSQL**, and **HSQL**.

dsname.desc

Parameter	Type	Default value
dsname.desc	String	

Sets an optional, human readable description of the data source.

> **Note**
> Any other parameter specific to the JDBC driver can be added to this configuration, as long as it is prefixed by the data source name.

> ***Example***
> Here is an example of the minimal configuration of an H2 database source, where the database is embedded and the database file stored into the *./database* sub-folder of the application's home directory. The database file name is *myds* and the connection user is *sa*.
>
> ```
> myds.type = h2
> myds.password = password
> ```
>
> This configuration is equivalent to the following:
>
> ```
> myds.url = jdbc:h2:./database/myds
> myds.login = sa
> myds.password = password
> ```
>
> Here is an example of the minimal configuration of a PostgreSQL data source:
>
> ```
> myds.url = jdbc:postgresql://localhost/mydatabase
> myds.login = arcadlogin
> myds.password = password
> ```

H2 Database encryption

The embedded H2 database can be encrypted to enhance the confidentiality of the data stored. Note that this encryption is not hard enough to be considered as a valid protection against a determined attacker. It may, however, be a piece of a global policy or as a minimalist response allowing to protect against simple menaces or delaying an attack.

Follow the subsequent steps to activate the encryption of your database. For this procedure, you need to use the Configuration tools for databases.

1. Stop the application.
2. Make a backup of the database with the *h2backup* command tool.
3. Delete the database (by default the file *.mv.h2 stored into the *./database* folder.
4. Change the configuration of the data source.
   a. Add `;CIPHER=AES` (without double quotes) at the end of the data source URL.
   b. Add an encryption password after the password used by the *sa* user to connect to the database. The two password must be separated by a space.

5. Restore the backup of the database with the *h2restore* command tool.
6. Restart the application.

It is recommended to use strong passwords, especially for the encryption one.

> ***Example***
> Based on the previous example, the modification of the configuration to encrypt the database should be as follows:
>
> ```
> myds.url = jdbc:h2:./database/myds;CIPHER=AES
>
> myds.password = password encryptionpassword
> ```

> **Important!**
> Enabling the encryption of the database may have an negative impact on the performances of the application.

TLS connections

To configure a TLS connection to the database, refer to the documentation of the target database and add the required parameter to the data source configuration. Do not forget to add the data source name prefix.

> ***Example***
> For example, to activate the TLS connection to the previous example of a PostgreSQL connection "myds", add the following parameters:
>
> ```
> myds.ssl = true
> myds.sslmode = verify-full
> myds.sslrootcert = ./ca.crt
> ```

> ***Reference***
> For more information, refer to the PostgreSQL documentation.

4 H2 Database

Configuration located in the *com.arcadsoftware.h2.cfg* file or in the section *[H2]* of the *osgi.cm.ini* file.

In this section, manage the parameters of the H2 database server. Using an embedded H2 database in production offers better performances and makes the activation of an H2 Database Server no longer necessary.

autostart

Parameter	Type	Default value
autostart	Boolean	false

Enables or disables the H2 server when this application starts. By default, this option is disabled (set to false).

remote

Parameter	Type	Default value
remote	Boolean	false

Enables or disables the remote connection. By default, this option is disabled (set to false). When this option is disabled (false), only connections from the localhost are accepted.

ssl

Parameter	Type	Default value
ssl	Boolean	false

If enabled (set to true), this parameter activates TLS over the TCP-IP connection to this server. By default, this option is disabled (set to false).

port

Parameter	Type	Default value
port	Integer	9092

Sets the TCP-IP port number the H2 database server listens on. The valid port number must be an integer between 1 and 65535. The port number must be free on this machine (i.e. not used by another server, including the TCP servers running in this application).

trace

Parameter	Type	Default value
trace	Boolean	false

If enabled (set to true), this parameter activates the trace log of the H2 Database server. By default, this option is disabled (set to false) This option limits the performance of the database server.

create

Parameter	Type	Default value
create	Boolean	false

If enabled (set to true), this parameter allows the H2 database server to automatically create a non-existing database container at first connection. By default, this option is disabled (set to false).

5 HTTP and SOCKS Proxy

Configuration located in the *config.ini* file.

In this section, configure the proxy connection, if the HTTP, FTP or any TCP-IP (SOCKS) connections require to use a proxy server on your network.

The configuration mainly depends on the type of protocol. If an authentication is required to access to the proxy server, refer to the corresponding feature configuration.

5.1 HTTP

http.proxyHost and **http.proxyPort**

Parameter	Type	Default value
http.proxyHost	String	
http.proxyPort	Integer	80

Sets the host name and the port number of the HTTP proxy server.

http.nonProxyHosts

Parameter	Type	Default value
http.nonProxyHosts	String	

Sets a list of hosts that should be reached directly, bypassing the proxy. The elements of the list must be separated by a vertical bar (**|**)' and can start or end with an asterisk (*****) for wildcards.

Any host matching one of this list is reached via a direct connection instead of a proxy.

5.2 HTTPS

https.proxyHost and **https.proxyPort**

Parameter	Type	Default value
https.proxyHost	String	
https.proxyPort	Integer	443

Sets the host name and the port number of the HTTPS proxy server.

If these parameters are not set, the HTTPS connection does not go through the Proxy server.

> **Note**
> The list of hosts set in the **http.nonProxyHosts** parameter can be used for HTTPS hosts too.

5.3 FTP

ftp.proxyHost and ftp.proxyPort

Parameter	Type	Default value
ftp.proxyHost	String	
ftp.proxyPort	Integer	80

Sets the host name and the port number of the FTP proxy server.

ftp.nonProxyHosts

Parameter	Type	Default value
ftp.nonProxyHosts	String	

Sets a list of hosts that should be reached directly, bypassing the proxy. The elements of the list must be separated by a vertical bar (**|**)' and can start or end with an asterisk (*****) for wildcards.

Any host matching one of this list is reached via a direct connection instead of a proxy.

5.4 SOCKS

socksProxyHost and socksProxyPort

Parameter	Type	Default value
socksProxyHost	String	
socksProxyPort	Integer	1080

Sets the host name and the port number of the proxy server used for TCP and UDP socket connection, other than HTTP, HTTPS and FTP.

6 Log

Configuration located in the *org.ops4j.pax.logging.cfg* file or in the section *[Log]*section of the *osgi.cm.ini* file.

The logger aggregates all the logs of the application into one unique file, using the java Log4j engine version 2 as a back-end. Configuring this logger is equivalent to configuring the Log4j engine, through properties prefixed by *log4j2*.

The Log4j configuration contains two elements:

- the loggers: they are associated to the Java code, allowing it to send messages,
- the appenders: they transpose these messages to different media.

6.1 Root logger

The **rootLogger** is at the top of all loggers hierarchy in which all other loggers are aggregated. It is associated to a log level which defines all types of messages stored through the "appenders".

The different levels of messages are:

- **error**: abnormal events caused by a compromised behavior or an issue during the execution of the program. The error level is the recommended level for production environment.
- **warn**: abnormal events related to a recoverable problem. They are not necessarily related to an error but they can lead to an error in the future *, and then may explain the cause of a following error*.
- **info**: normal events containing a detailed message about the global path of the execution through the program.
- **debug**: the debug message is more verbose than the info message as it describes the precise path through the program. The production of these messages may slow down the execution and display internal application data in the log files.

> **Note**
> The log levels are hierarchical, each level includes the upper levels. If you select the **info** level, the **error**, **warn** and **info** level messages are sent to the appenders.

There are two other levels:

- **fatal**: above the error level, the associated messages only concern critical errors. For AFS, this level is not used, and is equivalent to turning the log off.
- **trace**: this level can be used by some third-party modules. For AFS, this level is equivalent to the debug level.

To set the global log level, the **log4j2.rootLogger.level** option must be configured.

log4j2.rootLogger.level

Parameter	Type	Default value
log4j2.rootLogger.level	String	

The accepted values are any of the log level or the value **off** which turns off the logging.

6.2 Appenders

The appenders store the log messages in different media. Log4j version 2 provides a large set of appenders. The two most convenient ones are the **Rolling File Appender** and the **Console Appender**. But some other appenders can be interesting, such as the **SMTP appender** or the **Syslog Appender**.

The **Rolling File Appender** stores the log in the file system using a defined number of files with limited maximum sizes (rolling). This configuration is quite verbose, each set of properties is composed of several sub-properties. The *.type properties are used to clarify the meaning of the properties with the same prefix.

The configuration properties of this appender are the following, each one presented with its default value.

log4j2.appender.file.type

Parameter	Type	Default value
log4j2.appender.file.type	String	RollingFile

Sets the type of the appender.

log4j2.appender.file.name

Parameter	Type	Default value
log4j2.appender.file.name	String	File

Sets the name of the appender.

log4j2.rootLogger.appenderRef.file.ref

Parameter	Type	Default value
log4j2.rootLogger.appenderRef.file.ref	String	File

Enables this appender by associating it to the root logger.

log4j2.appender.file.layout.type

Parameter	Type	Default value
log4j2.appender.file.layout.type	String	PatternLayout

The layout uses a pattern defined layout. Other layouts exist to generate different formats of log file, such as JSON, HTML, CSV or Syslog formats.

log4j2.appender.file.layout.pattern

Parameter	Type	Default value
log4j2.appender.file.layout.pattern	String	%d %p %t %c - %m%n

Sets the pattern used where:

- **%c** includes the name of the sub-logger of the message.
- **%C** includes the java Class name of the issuer of the message.
- **%d{date-format}** includes the generation date of the message.
- **%ex{param}** or **%xEx{param}** explicitly includes the stack-trace of the exception associated with the message. By default, the full stack-trace is printed. Add **%ex{0}** to the pattern if you do not want the stack trace printed.
- **%l** includes the location of the caller that generated the message. This includes the full method name and the source code line number.
- **%L** includes only the source code line of the caller, if known.
- **%m** includes the log message.
- **%n** adds the platform dependent return carriage.
- **%N** includes a timestamp in nanoseconds.
- **%p** includes the log level.
- **%r** includes a timestamp since the start of the application, expressed in milliseconds.
- **%sn** includes a sequence number.
- **%T** outputs the ID number of the thread that generated the message.
- **%t** outputs the name of the thread that generated the message.

log4j2.appender.file.fileName

Parameter	Type	Default value
log4j2.appender.file.fileName	String	./logs/server.log

Sets the path to the current file.

log4j2.appender.file.filePattern

Parameter	Type	Default value
log4j2.appender.file.filePattern	String	./logs/server_%d {yyyy-MM-dd}_%i.log

Sets the pattern name of the archived log files. In the default pattern, **%d{date-pattern}** is replaced by a time stamp, and **%i** is replace by an incremental number. The archive is compressed if this pattern ends with the following extentions: *.zip*, *.gz*, *.bz2*, *.deflate*, *.pack200*or *.xz*.

log4j2.appender.file.policies.type

Parameter	Type	Default value
log4j2.appender.file.policies.type	String	Policies

States that the following properties are relative to the policy to use when a rollover occurs.

log4j2.appender.file.policies.size.type

Parameter	Type	Default value
log4j2.appender.file.policies.size.type	String	SizeBased TriggeringPolicy

Sets a policy based on the size of the log file.

log4j2.appender.file.policies.size.size

Parameter	Type	Default value
log4j2.appender.file.policies.size.size	String	10 MB

Creates an archive of the logger file as soon as the file size exceeds the chosen size.

log4j2.appender.file.strategy.type

Parameter	Type	Default value
log4j2.appender.file.strategy.type	String	Default Rollover Strategy

Sets the type of the rollover strategy.

log4j2.appender.file.strategy.max

Parameter	Type	Default value
log4j2.appender.file.strategy.max	Integer	7

Sets the maximum number of archive files to store. When the maximum number of archive files is reached, older files are removed.

log4j2.appender.file.strategy.compressionLevel

Parameter	Type	Default value
log4j2.appender.file.strategy.compressionLevel	Integer	0

Sets the compression level to use when the file name pattern ends with a compression file extension.

Setting this option to 0 disables the compression. The value is an integer between 1 (the fatest compression) and 9 (the higher compression rate).

The **Console Appender** allows to print the log to the system console, if the application is executed interactively.

log4j2.appender.console.type

Parameter	Type	Default value
log4j2.appender.console.type	String	Console

Sets the type of the appender.

log4j2.appender.console.name

Parameter	Type	Default value
log4j2.appender.console.name	String	Console

Sets the name of the appender.

log4j2.rootLogger.appenderRef.console.ref

Parameter	Type	Default value
log4j2.rootLogger.appenderRef.console.ref	String	Console

Enables this appender by associating it to the root logger.

log4j2.appender.console.layout.type

Parameter	Type	Default value
log4j2.appender.console.layout.type	String	PatternLayout

Sets the layout type for this appender.

log4j2.appender.console.layout.pattern

Parameter	Type	Default value
log4j2.appender.console.layout.pattern	String	%d %p %t %c - %m%n

Sets the format of the message printed on the console screen.

6.3 Other Loggers

Along with these parameters, there are specific configuration parameters for some modules of our applications. This is the case with the HTTP Server (Eclipse Jetty) which is quite verbose at the info and debug levels.

log4j2.logger.jetty.name

Parameter	Type	value
log4j2.logger.jetty.name	String	org.eclipse.jetty

Associates this logger to the Jetty HTTP server.

log4j2.logger.jetty.level

Parameter	Type	value
log4j2.logger.jetty.level	String	warn

Sets the specific level for this logger.

7 Properties

Configuration located in the *com.arcadsoftware.server.properties.cfg* file or in the section *[Properties]* of the *osgi.cm.ini* file.

In this section, manage the external storage used to store the internationalization files of the *Dynamic Editors.*

basedir

Parameter	Type	Default value
basedir	String	./files/properties

Sets the external folder path used to store the internationalization properties files. By default, the value is relative to the application's home directory.

> **Important!**
> This storage location is for an internal usage and should not be modified in a production environment.

watchdir

Parameter	Type	Default value
watchdir	Integer	120

Sets, in seconds, the delay used to track modifications in the **basedir** folder. By default, the value is set to 2 minutes.

8 REST

The HTTP server used to serve the REST Web-Services is configured with the following parameters. Both HTTP and HTTPS protocols are available and can be used simultaneously or independently.

The **HTTP server** communicates faster and is easier to configure, but it is recommended only for testing or diagnosis purposes.

The main configuration is located in the *com.arcadsoftware.server.restful.cfg* file or in the section *[REST]* of the *osgi.cm.ini* file.

port

Parameter	Type	Default value
port	Integer	5252

Sets the TCP-IP port number that the HTTP server listens on. The HTTP port number usually used is the number 80, but some firewalls may block any number below 1024, either because they are reserved or because another HTTP server already uses this port number on the same machine (i.e. including any TCP-IP servers run by this application). The valid port numbers must be between 1 and 65535. Setting the value to 0 (zero) disables the HTTP server. By default, the value is set to 5252.

> ***Warning!***
> The HTTP server is an unsecured server as the content of the communication is sent in plain text through the network.
>
> The **HTTPS server** encrypts its communication with TLS. This server is recommended for production. It requires the generation or the acquisition of keys and certificates from the Certificate authority. It is strongly recommended to set a zero value to this parameter and define the following ones.

portssl

Parameter	Type	Default value
portssl	Integer	

Sets the TCP-IP port number that the HTTPS server listens on. The HTTPS port number is generally 443, but it is recommended to set a port number above 1024 so that it is not blocked by the machine firewall, not reserved for another protocol and harder to find by a malicious program. The valid port number must be between 1 and 65535. Setting the value to 0 (zero) disables the HTTP server. The port number must be free on this machine. If the HTTP port is enabled the HTTPS portssl must be set to a different value.

keystore

Parameter	Type	Default value
keystore	String	

Sets the local path to the keystore containing the private key used to secure the HTTPS connections.

keystorepwd

Parameter	Type	Default value
keystorepwd	Encoded string	

Sets the password required to open the **keystore**. This value can be encrypted.

keytype

Parameter	Type	Default value
keytype	String	JKS

Sets the type of keystore used. The accepted value depends on the JVM capabilities. Most of the JVM implementations support the native **JKS** type and the **PKCS12** keystore types. By default, the value is JKS.

keyalias

Parameter	Type	Default value
keyalias	String	

Sets the alias of the private key used by the HTTPS connection.

keypwd

Parameter	Type	Default value
keypwd	Encoded string	

Sets the password protecting the private key. This value can be encrypted.

clientauth

Parameter	Type	Default value
clientauth	Boolean	

If true, the HTTP client requires to be authenticated by a public key. This public key must be trusted by one of the certificates included in the **truststore**parameter below.

truststore

Parameter	Type	Default value
truststore	String	

Sets the local path to the keystore containing the public certificates used by the HTTP clients. This option is required only if the **clientauth** parameter is set to true.

truststorepwd

Parameter	Type	Default value
truststorepwd	Encoded string	

Sets the password needed to open the **truststore**. This value can be encrypted. This option is required only if the **clientauth**parameter is set to true.

truststoretype

Parameter	Type	Default value
truststoretype	String	JKS

Sets the type of truststore used. The accepted value depends on the JVM capabilities, most of the JVM implementations support the native **JKS** type and the **PKCS12** keystore types. By default, the value is JKS.

Common configuration:

domainname

Parameter	Type	Default value
domainname	String	

Sets the list of names, separated by spaces, by the client to reach this HTTP or HTTPS server. These names may be the DNS names of this host or any reverse proxy names.

Represent a list of domain names corresponding to this machine hostname, separated by spaces, used by the client to reach this HTTP or HTTPS server. These names may be the DNS names of this host or any reverse proxy names which are targeting the actual host of this application. The default value of this parameter is the actual host name as seen by the JVM running this application. Depending on the configuration of Ethernet interface of the machine, this default host name may not be accurate.

This parameter is used to set the value of *Access-Control-Allow-Origin* HTTP header.

author

Parameter	Type	Default value
author	String	ARCAD Software

This is an informal parameter used to self document the server. By default, this option is set to ARCAD Software.

name

Parameter	Type	Default value
name	String	

This is an informal parameter used to self document the server. By default, this option is set to the ARCAD Product name.

version

Parameter	Type	Default value
version	String	

This is an informal parameter used to self document the server. By default, this option is set to the ARCAD Product current version.

owner

Parameter	Type	Default value
owner	String	

This is an informal parameter used to self document the server. The value of this parameter can be set to the customer name.

branding.bundle

Parameter	Type	Default value
branding.bundle	String	

If set, this parameter must be equivalent to the symbolic name of an installed bundle.

> **Important!**
> The *branding.bundle* option, if set, overrides the values set in the **author**, **name** and **version** options.

log.disabled

Parameter	Type	Default value
log.disabled	Boolean	true

If enabled (set to true), this parameter disable the HTTP logging. This can avoid a large number of log content especially at the INFO or DEBUG level. By default, this option is enabled (set to true).

Some other properties located in the *config.ini* may impact the REST server configuration:

restlog.file

Parameter	Type	Default value
restlog.file	String	./restfulports.log

Sets the path to a log file in which the current port number of the HTTP and HTTPS server is stored (for information only). By default, the value is *./restfulports.log*. Setting an empty value disables the creation of such file.

com.arcadsoftware.rest.port

Parameter	Type	Default value
com.arcadsoftware.rest.port	Integer	

Modifies the default value of the HTTP port number, i.e. 5252, if there is no specific configuration provided. Setting this value to 0 disables the HTTP server.

9 REST Connection

Configuration located in the *com.arcadsoftware.server.restful.connection.cfg* file or in the section *[REST Connection]* in the *osgi.cm.ini* file.

In this section, define the user authentication properties required to access the application web-services.

cache

Parameter	Type	Default value
cache	Integer	30

Sets the cache duration, in seconds, used to store the users credential obtained after a first successful authentication.

inactive

Parameter	Type	Default value
inactive	Boolean	false

If enabled (set to true), this parameter disables all authentication to the application's web-services. All services requiring an authentication will not be accessible. By default, this option is disabled (set to false).

realm

Parameter	Type	Default value
realm	String	arcad-evolution

Sets the HTTP BASIC realm used to authenticate the connections to the web-services.

10 REST Connection LDAP

Configuration located in the *com.arcadsoftware.server.restful.connection.ldap.cfg* file or in the section *[REST Connection LDAP]* section of the *osgi.cm.ini* file.

In this section, manage the connection bridge between the application's web-services and an LDAP server, including the LDAP connection parameters and other parameters allowing user importation into the application.

server

Parameter	Type	Default value
server	String	

Defines the LDAP server address and port number (if different from default port number 389 or 636).

The value of this parameter is equal to "hostname" or "hostname:port" where *hostname* is the actual host name or IP address of the LDAP server and *port* is the TCP-IP port number used to serve the LDAP protocol interface.

servers

Parameter	Type	Default value
servers	String	

Like the **server** parameter, this parameter defines the LDAP server access and port number, but this parameter accept multiple servers declaration, each one separated by a space character, allowing to support a cluster of LDAP server used as a *failover* server set.

host

Parameter	Type	Default value
host	String	

This parameter is the legacy parameter used to define the LDAP server access and is replaced by the **server** and **servers** parameters. It sets the LDAP server host DNS or IP address.

> **Note**
> The parameters **server**, **servers** and **host** may be combined to define a set of failover servers. In that case, the first server in the list will be the one specified by the parameter **host**, followed by the one in **server** and then the server list of **servers**. However, you should avoid repeating the same "hostname:port" couple in different parameters' position. In other words, if you want to manage a cluster of LDAP servers use **servers**. If you have a unique server connection, use either **server** or **host**, but not both.

port

Parameter	Type	Default value
port	Integer	389 or 636

Sets the default TCP port number associated with the given list of LDAP servers. By default, this port is equal to 389 for LDAP connection, or to 636 if the TLS connection parameters are defined, assuming an LDAPS connection.

reorderfailoverlist

Parameter	Type	Default value
reorderfailoverlist	Boolean	false

Specifies whether the list of servers used by this failover server set should be re-ordered when a failure is encountered while attempting to establish a connection. By default, the original order will be preserved. If enabled (set to true), then a failed attempt to establish a connection to the server at the beginning of the list will cause that server to be moved to the end of the list so that it will be the last server/set tried on the next attempt. By default, this option is disabled (set to false).

connection.pool.initial.size

Parameter	Type	Default value
connection.pool.initial.size	Integer	1

Defines the number of connections to establish initially when the LDAP connection pool is created. It must be greater or equal to one.

connection.pool.max.size

Parameter	Type	Default value
connection.pool.max.size	Integer	equals to **connection.pool.initial.size**

Define the maximum number of connections that should be maintained in the pool. It must be greater than or equal to the initial number of connections.

Whenever a connection is needed, the LDAP Connection pool first checks if there is a connection that has already been established but is not currently in use. If so, then that connection is used.

If there are not any unused connections that are already established, then the pool determines if it has yet created the maximum number of connections. If not, then it immediately creates a new connection and uses it.

If the pool has already created the maximum number of connections, then the pool waits for a period of time (as indicated by the **timeout** parameter) for an in-use connection to be released back to the pool. If no connection is available after the specified waiting time (or there may not be any waiting time), then the pool automatically creates a new connection to use it.

connection.pool.available.goal

Parameter	Type	Default value
connection.pool.available.goal	Integer	equals to **connection.pool.initial.size**

Specifies the goal for the minimum number of available connections that the pool should try to maintain for immediate use. If this goal is greater than zero, then the health checking process attempts to create enough new connections to achieve this goal. A value less than or equal to zero indicates that the pool should not try to maintain a minimum number of available connections.

connection.pool.max.age

Parameter	Type	Default value
connection.pool.max.age	Integer	120000

Specifies the maximum length of time in milliseconds that a connection in the pool may be established before it should be closed and replaced with another connection. A value of zero indicates that no maximum age should be enforced.

connection.failover.max.age

Parameter	Type	Default value
connection.failover.max.age	Integer	60000

Specifies the maximum connection age, in milliseconds, that should be used for connections that were created in order to replace defunct connections. Define a custom maximum connection age for these connections to allow them to be closed and re-established more quickly, for a potentially quicker failback to a normal state.

If you want to use the feature, then the maximum age for these connections should be long enough to allow the problematic server to become available again under normal circumstances (e.g., it should be long enough to allow for at least a shutdown and restart of the server, and also for potentially performing routine maintenance while the server is offline, or for a chance for an administrator to restore the server that went down).

timeout

Parameter	Type	Default value
timeout	Integer	5000

Specifies the maximum length of time, in milliseconds, to wait for a connection to become available when trying to obtain a connection from the pool. Set the parameter to 0 to indicate that the connection pool should not block at all if no connections are available and that it should create a new connection as soon as it is required.

busy.retry

Parameter	Type	Default value
busy.retry	Integer	1

Sets the number of retries during binding operations, when the server returns a *"Busy"* error. The maximal number of retries allowed is 12 and the client has to wait 2 seconds between each retry.

ssl.type

Parameter	Type	Default value
ssl.type	String	

When set, this parameter enables the secured connection with the LDAP server (using TLS). Accepted values depend on the JVM implementation and on the requirement of the LDAP server, they can be:**TLS**, **TLSv1**, **TLSv1.1**, **TLSv1.2**, **TLSv1.3**.

> ***Example***
>
> To configure the TLS support, only the following parameters are generally required:
>
> ssl.type = TLS
> ssl.truststore.path = ... with the local path to the JKS truststore containing the server certificate if it is not a CA generated certificate.
> ssl.truststore.pwd = ... with the password required to open the provided truststore.

ssl.prefered.protocol

Parameter	Type	Default value
ssl.prefered.protocol	String	TLSv1.3

Defines the preferred TLS protocol to be used in TLS negociation with the server if the **ssl.type** parameter is set to "TLS".

ssl.protocols.enabled

Parameter	Type	Default value
ssl.protocols.enabled	String	TLSv1.2 TLSv1.3

Define the list of TLS protocols open to negotiation with the server, if the **ssl.type** parameter is set to "TLS". Each value is separated by a white space character.

ssl.protocols.disabled

Parameter	Type	Default value
ssl.protocols.disabled	String	

Defines the list of TLS protocol to remove from the negotiation with the server, if the **ssl.type** parameter is set to "TLS". Each value is separated by a white space character. This parameter is not useful if the **ssl.protocols.enabled** parameter is defined, as only an enabled protocol will be accepted anyway.

ssl.provider

Parameter	Type	Default value
ssl.provider	String	

If set, define the TLS implementation provider name to be used for this connection. Acceptable values depend on the JVM implementation currently used. If this parameter is not set, the first provider declared in the JVM is used. Acceptable values may be: BC, SunJSSE or IBMJSSE2.

ssl.ciphersuites.enabled

Parameter	Type	Default value
ssl.ciphersuites.enabled	String	

Defines the list of cipher algorithm suites explicitly authorized during the TLS negotiation with the server. The values in this list must be separated by a white space character and are case-insensitive. The complete list of the accepted cipher suites depends on the JVM implementation. You may have to refer to the security documentation of the used JVM to find the exact list, which also depends on the provider used(e.g. the currently used list of cipher suites used by the SunLSSE provider).

ssl.ciphersuites.disabled

Parameter	Type	Default value
ssl.ciphersuites.disabled	String	

Defines the list of cipher algorithm suites to disable during the TLS negotiation with the server. Each value is separated by a white space character. This parameter is not useful if the **ssl.ciphersuites.enabled** parameter is defined, as only the enabled cipher suites will be accepted anyway.

ssl.random

Parameter	Type	Default value
ssl.random	String	

Sets the random algorithm used in the TLS handshake. Acceptable values depend on the JVM implementation and the current values are**SHA1PRNG**, **NativePRNG** and **DRBG**. By default, the default value of the JVM implementation is used.

ssl.start

Parameter	Type	Default value
ssl.start	Boolean	

If set to true, this parameter enables the **Start TLS** protocol, which, if the server allows it, uses the same TCP port number for a clear and secure connection.

ssl.truststore.path

Parameter	Type	Default value
ssl.truststore.path	String	

Sets the local path to the trusted certificate store containing the LDAP Server public certificate. If this parameter is not set, the JVM default truststore is used.

ssl.truststore.pwd

Parameter	Type	Default value
ssl.truststore.pwd	Encoded string	

Sets the password required to open the **ssl.truststore.path**. This value can be encrypted.

ssl.truststore.type

Parameter	Type	Default value
ssl.truststore.type	String	JKS

Sets the type of the truststore. Accepted values depend on the JVM capabilities. Most JVM implementations support the native **JKS** type and the **PKCS12** keystore types.

ssl.truststore.algorithm

Parameter	Type	Default value
ssl.truststore.algorithm	String	

Sets the algorithm used to manage the **X.509** certificates from the truststore. Accepted values depend on the JVM implementation. Current accepted value are **SunX509**, **IbmX509** or **PKIX**. By default, the JVM default value is used.

ssl.keystore.path

Parameter	Type	Default value
ssl.keystore.path	String	

Sets the local path to the private keystore containing the LDAP Client private key. By default, the JVM default keystore is used.

ssl.keystore.pwd

Parameter	Type	Default value
ssl.keystore.pwd	Encoded string	

Sets the password required to open the keystore file. This value can be encrypted.

ssl.keystore.type

Parameter	Type	Default value
ssl.keystore.type	String	JKS

Sets the type of the keystore. Accepted values depend on the JVM capabilities. Most of the JVM implementation support the native **JKS** type and the **PKCS12** keystore types.

ssl.keystore.algorithm

Parameter	Type	Default value
ssl.keystore.algorithm	String	

Sets the algorithm used to manage the **X.509** certificates from the keystore. Accepted values depend on the JVM implementation. Current accepted values are **SunX509**, **IbmX509**or **PKIX**. By default, the JVM default value is used, e.g. SunX509 with the ORACLE JVM or ibmX509 with the IBM one.

ssl.keystore.keypwd

Parameter	Type	Default value
ssl.keystore.keypwd	String	same value as **ssl.keystore.pwd**

Sets the password required to access to the keys stored into the keystore. By default the same password used to open the keystore is used here, but if this keystore is also used to store other key materials, a different password is recommended.

bind.type

Parameter	Type	Default value
bind.type	String	simple

This parameter define the method used to bind the user into the LDAP server. The accepted values are:

- **simple** (the default method): this method performs an LDAPv3 simple bind operation, which authenticates using a bind DN and password.
- **plain**: this method performs an SASL PLAIN bind request using the authentication ID and password. Even if compatible with SASL, this method will not use any external authorization token to bind to the LDAP server.
- **scram-sha-1**: this method performs an SASL bind request using the salted challenge-response authentication mechanism (SCRAM) with SHA-1 digest algorithm as described in RFC5802.
- **scram-sha-256**: this method performs an SCRAM SASL bind request with SHA-256 algorithm as described in RFC 7677.
- **scram-sha-512**: this method performs an SCRAM SASL bind request with SHA-512 algorithm and an HmacSHA512 MAC algorithm.
- **cram-md5**: this method performs an SASL CRAM-MD5 bind request allowing to authenticate over an insecure channel without exposing the credentials (although it requires that the server has access to the clear-text password). That is, the **dn.login.pattern** parameter should be set either to "dn:%s" with the distinguished name of the target user, or "u:%s" to use the username to authenticate. If the "u:" form is used, then the mechanism used to resolve the provided username to an entry may vary from server to server.
- **digest-md5**: this method performs an SASL DIGEST-MD5 bind request, as described in RFC 2831. This mechanism can be used to authenticate over an insecure channel without exposing the credentials (although it requires that the server has access to the clear-text password). It is similar to CRAM-MD5, but provides better security by combining random data from both the client and the server. It also allows for greater security and functionality, including the ability to specify an alternate authorization identity and the ability to use data integrity or confidentiality protection. Note that the current implementation does not exploit the alternate authorization identity, and can only target the server default realm.

- **gssapi**: this method performs a SASL GSSAPI bind request as described in RFC 4752. It makes it possible to authenticate to a directory server using Kerberos V. This implementation uses the Java Authentication and Authorization Service (JAAS) to perform all Kerberos processing. This framework requires a configuration file to indicate the underlying mechanism to be used, this configuration file may change according to the used JVM.

Note that each of these methods require a specific server configuration and specific changes in the following parameters, especially the **dn.login.pattern** parameter.

realm

Parameter	Type	Default value
realm	String	

Specifies the realm into which the user should authenticate. If this is not provided, an attempt is made to determine the appropriate value from the system configuration.

kerberos.kdc

Parameter	Type	Default value
kerberos.kdc	String	

Specifies the IP address or resolvable name for the Kerberos key distribution center for the **gssapi** bind type. If this is not provided, an attempt is made to determine the appropriate value from the system configuration.

dn.base

Parameter	Type	Default value
dn.base	String	

Sets the LDAP base distinguished name to search users from. The value must be a valid LDAP distinguished name representing the LDAP node in which the user is allowed to connect to this application.

This parameter is required only to perform search operations in the LDAP, like user import and login verification. These operations are disabled if this parameter is not set.

search.user.dn

Parameter	Type	Default value
search.user.dn	String	

Sets the distinguished name of a permanent LDAP connection used to bind other users and manage the import operations. This LDAP user must be permanently accessible to the application server, if not no LDAP user connection will be possible.

This connection allows to search the login of any user trying to connect to the application into the LDAP tree before trying to authenticate them. To do so, the **attribute.login** parameter must be defined, and the **dn.login.pattern** parameter must not be used.

This mode of authentication of the application users, and the **search.user.*** parameter are optional, it may lower performance compared to a direct binding of the LDAP user and may require to permanently store the corresponding LDAP user password into the configuration files.

search.user.password

Parameter	Type	Default value
search.user.password	String	

Sets the password used by the corresponding **search.user.dn** parameter.

search.user.sso

Parameter	Type	Default value
search.user.sso	String	

Makes it possible to exploit an existing token or method to authenticate the permanent connection to the LDAP, defined by the **search.user.dn** and **search.user.password** parameters, allowing this connection, and only to this one, to perform a *"single sign-on"* operation.

The accepted values are the following:

- **kerberos**: the permanent connection is bound using a GSSAPI protocol. In that case, if an opened kerberos token is accessible through the operating system, the application uses it and the **search.user.dn** and **search.user.password** parameters are not required.

> **Note**
> This method relies on the JAAS implementation of Kerberos and may require a specific configuration of the JVM.

- **external**: use an SASL EXTERNAL bind request implementation as described in RFC 4422. The EXTERNAL mechanism is used to authenticate using information that is available outside of the LDAP layer (e.g., a certificate presented by the client during TLS or StartTLS negotiation). Depending on the External mechanism, a **search.user.dn** may still be required to perform the bind operation.

attribute.login

Parameter	Type	Default value
attribute.login	String	

Sets the LDAP attribute associated with the user login used by the application. Accepted values depend on the LDAP server.

This parameter is only required to perform user importation from the LDAP server. But it may also be used as the replacement of the **dn.login.pattern** parameter.

dn.login.pattern

Parameter	Type	Default value
dn.login.pattern	String	

This parameter transforms the user login into an acceptable LDAP identifier for the bind request. The value of this parameter should contain the "%s" placeholder which will be replaced by the actual user login. Its value may depend on the **bind.type** used and the LDAP server capabilities, but current usage are as the following:

- *%s* if the login is directly equal to the required binding identifier. This may be the required pattern to authenticate to an Active Directory server with the same identifier used to connect to the Windows account (the complete login including the domain name, e.g. "login@domain.com" or "NTDOMAIN\login").
- or *%s@domain.com* to simply use as a login the short version of the UserPrincipalName attribute in Active Directory.
- or even *uid=%s,ou=UserOrganization,dc=domain,dc=com* to use any part of the distinguished names as login.

If this parameter is not set, then if **dn.login** and **dn.base** are set, the binding identifier will be build as the result of the following pattern "***dn.login***= %s ,***dn.base***". For exemple, if **dn.login**=sAMAccountName and **dn.base**=dc=domain,dc=com, the default pattern will be "sAMAccountName=%s,dc=domain,dc=com". If **dn.login** is not set, then the default pattern will be "%s", i.e. the user login is directly used as the binding identifier.

A "%S" placeholder, with an upper-case S, may be used to automatically transform all "login" in upper-case before using them to bind to the LDAP server.

> ***Reference***
> For more information about the syntax of LDAP distinguished names , refer to the LDAP DN documentation.

attribute.password

Parameter	Type	Default value
attribute.password	String	userPassword

Sets the LDAP attribute associated to store the user password. Accepted value depend on the LDAP implementation, it may be possible that the LDAP Server do not authorize the modification of the user password through an LDAP connection.

> **Note**
> This parameter is only used when the **password.modify** parameter is set to true.

password.modify

Parameter	Type	Default value
password.modify	Boolean	

If set to true, allows the users are allowed to change their LDAP user account password through this application's services. To do so, the**dn.password**parameter must be set to a non-empty value.

> **Important!**
> In a production environment, it is not recommended to allow the end users to change their password through an external application. Moreover, only LDAP passwords stored in clear into the LDAP server can be modified through the LDAP connection.

userimportenabled

Parameter	Type	Default value
userimportenabled	Boolean	false

Define if the import of LDAP user in the database of the application is enabled. By default, this option is disabled (set to false). To use this function, it is also required to define the parameters **dn.login**, **dn.base** and **ldapmap**.

login.casesensitive

Parameter	Type	Default value
login.casesensitive	Boolean	true

If enabled (set to true), the LDAP user logins are case-sensitive. By default, this option is enabled (set to true).

ldapmap.*x*

Parameter	Type	Default value
ldapmap.	String	

Every parameter starting with the prefix **ldapmap** defines the LDAP attributes that are associated with the corresponding attribute ("x" in this example) of the user entity in the database of the application. The values of these parameters accept multiple LDAP attribute names, separated by a white space. This list is a priority list: the first LDAP attribute existing in the corresponding LDAP node is used. For the user importation feature, a default constant value can be specified at the end of the list, using a double-quoted string. In this case, the imported user attribute in the database of the application is filled with the content of the quoted string, if no previous LDAP attribute is found in the corresponding LDAP node.

> ***Example***
> The following LDAP configuration is an example for a Microsoft Active Directory LDAP server:

```
host = < your server DNS name>
port = 389
dn.attribute= sAMAccountName
dn.login.pattern = %s@<local domain name>
user.forcebinding = true
dn.base = <depends on your LDAP tree>
```

Reference

For more information, refer to the Microsoft documentation: https://docs.microsoft.com/en-us/windows/win32/adschema/active-directory-schema.

11 REST Connection Local

Configuration located in the *com.arcadsoftware.server.restful.connection.local.cfg* file or in the section *[REST Connection Local]* of the *osgi.cm.ini* file.

In this section, manage the web-services credentials stored in the database of the application. This module uses the database to store the login and password of users. Most of these configuration parameters are related to the constraints applied to the new password complexity.

> **Important!**
> Passwords are never written in the database. They are hashed, and only the hash of the password is stored.

lockcount

Parameter	Type	Default value
lockcount	Integer	5

Sets the maximum number of connections attempts allowed before user accounts are locked.
The minimum value accepted is 1, which means that users' accounts are locked as soon as they try to log in with the wrong password. On an SOA environment where the connection may come from a browser, a value lower than 3 may cause to lock the users too quickly.

pwddelay

Parameter	Type	Default value
pwddelay	Integer	3650 (10 years)

Sets, in days, the duration of a new password. This option sets the maximum validity of passwords. When a password reaches this limit, the user is prompted to change it.

admin.uid

Parameter	Type	Default value
admin.uid	List of integers	

Sets the internal user ID associated to the account, which must be unlocked after a specified delay (set in the **admin.unlock.delay** option below). Each ID must be separated by a white space.

All of these users are automatically unlocked every time the server program restarts, even if the delay is set to **0**.

admin.unlock.delay

Parameter	Type	Default value
admin.unlock.delay	Integer	0

Sets, in minutes, the automatic unlocking of locked admin users. Set the list of users in the **admin.uid** option above.

casesensitive

Parameter	Type	Default value
casesensitive	Boolean	true

If enabled (set to true), the user logins are case-sensitive. By default, this option is enabled (set to true).

password.min

Parameter	Type	Default value
password.min	Integer	0

Sets the minimum length of new passwords.

> **Important!**
> It is strongly recommended to set, at least, the **password.min** parameter to ensure a minimum degree of security in a production environment.

password.max

Parameter	Type	Default value
password.max	Integer	0

Sets the maximum length of new passwords. To disable the constraint, set this option to 0.

password.mindigit

Parameter	Type	Default value
password.mindigit	Integer	0

Sets the minimum number of digit characters (0 to 9) that must be included in new passwords.

password.minalpha

Parameter	Type	Default value
password.minalpha	Integer	0

Sets the minimum number of alphabetic characters (a to z and A to Z) that must be included in new passwords.

password.minnonalpha

Parameter	Type	Default value
password.minnonalpha	Integer	0

Sets the minimum number of non alpha-numeric characters (any character other than a to z, A to Z, and 0 to 9) that must be included in new passwords.

password.minchar

Parameter	Type	Default value
password.minchar	Integer	0

Sets the minimum number of different characters that must be included in new passwords.

password.distlogin

Parameter	Type	Default value
password.distlogin	Integer	0

Sets the minimum Levenshtein distance between a new password and the user login. To disable the constraint, set this option to 0.

password.distold

Parameter	Type	Default value
password.distold	Integer	0

Sets the minimum Levenshtein distance between a new password and the previous password. To disable the constraint, set this option to 0.

This constraint is only applied when users are prompted to change their own password. When users set an new password in the administration panel of the application, this constraint does not apply.

password.charlogin

Parameter	Type	Default value
password.charlogin	Integer	0

Sets the maximum number of characters from a new password that can be used in the login, regardless of their position. To disable the constraint, set this option to 0.

password.charold

Parameter	Type	Default value
password.charold	Integer	0

Sets the maximum number of characters from an old password that can be used in the new password, regardless of their position. To disable the constraint, set this option to 0.

This constraint is only applied when users are prompted to change their own password. When users set an new password in the administration panel of the application, this constraint does not apply.

password.minlowercase

Parameter	Type	Default value
password.minlowercase	Integer	0

Sets the minimum number of lower case characters that must be used in the new password. Setting this option to 0 disables the constraint.

password.minuppercase

Parameter	Type	Default value
password.minuppercase	Integer	0

Sets the minimum number of upper case characters that must be used in the new password. Setting this option to 0 disables the constraint.

password.minrepeated

Parameter	Type	Default value
password.minrepeated	Integer	0

Sets the maximum number of accepted characters consecutively repeated. This constraint is case sensitive (for example, *"AaaAbaabc"* contains 4 repeated *"a"*). Setting this option to 0 disables the constraint.

password.blacklist

Parameter	Type	Default value
password.blacklist	Path	./configuration/blacklist.txt

Sets the file path leading to a text file containing a list of forbidden passwords. This file must contain one password per line and any of the password listed there are rejected during the password complexity check.

> **Note**
> All the constraints available may lead to an incoherent set of constraints, or at least to a very complex set of constraints, especially when the **password.max** constraint is enabled.

12 Security and cryptography

Configuration located in the *config.ini* file.

In this section, manage the set of parameters of the general encryption algorithm provided by the Extract Server. There are three types of encryption algorithms used in the applications:

- **fog**: reversible and fast encryption. This encryption does not provide protection against attacks and there are no parameters associated with this option.
- **cypher**: reversible and robust encryption. The robustness of this algorithm is adjustable through the different elements that are used in the encryption algorithm. This encryption includes the following elements:
  - A *master key* defining the key material used to encrypt the data.
  - A *salt*, which is a random string, different for all encryptions. It is added to the encrypted data to avoid identification or recurrent patterns in the encrypted data.
  - An *initialization vector (IV)*, which is another random string used to initialize the encryption algorithm, starting at a different position every time data is encrypted.
  - A number of *iterations* of the algorithm recursively applied on itself.
- **hash**: irreversible encryption. It also uses a *salt* and a number of *iteration* to complicate the result of the encryption.

> **Note**
> The higher the following parameters value are, the more robust the corresponding algorithm is. However, setting higher values slows down the application and may impact its performance.

> ***Warning!***
> Setting lower values than the default ones may impact the security of the application.

com.arcadsoftware.masterkey.path

Parameter	Type	Default value
com.arcadsoftware.masterkey.path	File path	

If set, this parameter must point to an existing file containing the binary representation of the master key. This file content must not be altered in any way after the first start of the application.

If no file path is defined, or if the file path points to a file that does not exist, the master key will be initialized by one of the following parameters.

> ***Warning!***
> This parameter must not be changed after the first start of the application, or all encrypted data will be lost, and the execution of the application will be heavily impacted.

com.arcadsoftware.masterkey.fog

Parameter	Type	Default value
com.arcadsoftware.masterkey.fog	Encrypted string	

Sets the value of the master key used to encrypt data, using the **fog** algorithm. This parameter is optional, but should be set during the installation of any product, before the first run of the application.

If this parameter is not defined, the master key value may be set by the following parameter.

> ***Warning!***
> This parameter must not be changed after the first start of the application, or all encrypted data will be lost, and the execution of the application will be heavily impacted.

com.arcadsoftware.masterkey

Parameter	Type	Default value
com.arcadsoftware.masterkey	String	

Sets the value of the master key used to encrypt data. The key itself is defined in clear text.

In a production environment, either the **com.arcadsoftware.masterkey.fog** parameter or the **com.arcadsoftware.masterkey** parameter should be used, depending on your installation. Setting the master key in its clear textual value is not recommended in production environment.

> **Important!**
> This parameter is optional. If not set, a default value is randomly generated using different elements of the current application, including the path of the application and other contextual elements which are not documented. This default value ensures that two different programs do not share the same master key. However, if the context of execution of the application changes, it impacts all the

encrypted data.

Warning!
This parameter must not be changed after the first start of the application, or all encrypted data will be lost, and the execution of the application will be heavily impacted.

com.arcadsoftware.salt.min.size

Parameter	Type	Default value
com.arcadsoftware.salt.min.size	Integer	8

Sets the minimum size of the salt added to the encrypted and hashed data. The value is an integer between 4 and 128 bytes.

com.arcadsoftware.salt.size.variation

Parameter	Type	Default value
com.arcadsoftware.salt.size.variation	Integer	16

Sets the amplitude of the variation of the size of the salt added to the encrypted and hashed data. The value is an integer between 4 and 64 bytes. The size of the salt data changes for each encryption between min and min+ variation sizes.

com.arcadsoftware.iv.min.size

Parameter	Type	Default value
com.arcadsoftware.iv.min.size	Integer	10

Sets the minimum size of the vector used to initialize the encryption algorithm. The value is an integer between 4 and 16 bytes.

com.arcadsoftware.iv.size.variation

Parameter	Type	Default value
com.arcadsoftware.iv.size.variation	Integer	6

Sets the variation of the size of the initialization vector. The value is an integer between 0 and 12 bytes.

com.arcadsoftware.hash.min.iterations

Parameter	Type	Default value
com.arcadsoftware.hash.min.iterations	Integer	8675

Sets the minimum number of repetitions of the hashing algorithm. The value is an integer between 100 and 10,000,000.

com.arcadsoftware.hash.iterations.variation

Parameter	Type	Default value
com.arcadsoftware.hash.iterations.variation	Integer	1892

Sets the variation of the number of repetitions of the hashing algorithm. The value is an integer between 100 and 10,000,000.

com.arcadsoftware.cypher.min.iterations

Parameter	Type	Default value
com.arcadsoftware.cypher.min.iterations	Integer	8187

Sets the minimum number of repetitions of the ciphering algorithm. The value is an integer between 100 and 10,000,000.

com.arcadsoftware.cypher.iterations.variation

Parameter	Type	Default value
com.arcadsoftware.cypher.iterations.variation	Integer	1607

Sets the variation of the number of repetitions of the ciphering algorithm. The value is an integer between 100 and 10,000,000.

> **Note**
> All parameters other than the master key can be changed at any time as their modification does not impact the data already encrypted.

13 TELNET Console

Configuration located in multiple files.

This console allows a remote access to the OGSi console through the TELNET protocol.

> ***Warning!***
> The TELNET protocol is unsecured and should not be used in production environment.

Configuration located in the *org.console.telnet.cfg* file or is the section *[TELNET Console]* of the *osgi.cm.ini* file:

enabled

Parameter	Type	Default value
enabled	Boolean	false

Enables or disables the TELNET console server. By default, this option is disabled (set to false).

port

Parameter	Type	Default value
port	Integer	

Sets the TCP-IP port number used by the TELNET server. The valid port number is a value between 1 and 65535. The port number must be free on this machine (i.e. not used by another server, including the TCP servers run by this application).

host

Parameter	Type	Default value
host	String	

Sets the host, or the Ethernet interface IP address the TELNET Server listens on. Using*localhost*limits the connection to the loop-back interface, the local machine.

Configuration located in the *config.ini* file:

osgi.console

Parameter	Type	Default value
osgi.console	Integer or String *<host>:<port>*	

Replaces the above configuration section and enables the TELNET Console. The value can be the TCP port number used by the TELNET Server (refer to the **port** parameter above) or a string *<host>:<-port>*, where both parts are equivalent to the parameters mentioned above.

osgi.console.useConfigAdmin

Parameter	Type	Default value
osgi.console.useConfigAdmin	Boolean	true

Enables or disables the configuration of the OSGi remote console. By default, this option is enabled (set to true). If disabled (set to false), the *config.ini* configuration file is ignored.

14 Web

Configuration located in the *com.arcadosftware.server.web.cfg* file or in the section *[Web]* of the *osgi.cm.ini* file.

In this section, configure the static web resource access of this application's server. These parameters are defined to work with the web front-end and should not be modified in production environment, except the **disabled** parameter.

disabled

Parameter	Type	Default value
disabled	Boolean	false

If enabled (set to true), this parameter completely disables the web resources access on this server. By default, this option is disabled (set to false).

secure

Parameter	Type	Default value
secure	Boolean	false

If enabled (set to true), the static Web resources is put in the web-server part, which requires a user authentication before to access to the resources. By default, this option is disabled (set to false). As the Web front-end program generally offers in the interface to manage the authentication of the user (i.e. a connection dialog), the static resources themselves must be accessible without a user authentication. It depends on the way the web front-end program is implemented and should not be modified in the production environment.

webroot

Parameter	Type	Default value
webroot	String	first mounted folder

Generally, the first folder defined as a container of static web resources is defined as the default web path (i.e. the root in the URL path, "/"). If more than one folder is defined, this default path may be randomly selected depending on the installation.

Setting an empty value to this parameter disable the "root path" of any static web resource container.

15 License Key

The installation of a license key may be required to run the application. The exact content of this key depends on the application, but it will validate the installation of the application.

Once installed and configured, the application will be able to start running, but to perform any license related actions, or to extends the default time on execution of the application, you have to install the license key provided by DOT Software. To generate this key the support line may have to ask you which are the **system parameters** of the application.

A license key is either an encoded text such as the following text. Depending on the application and the key itself, the text may be longer, or the license key may be a binary file named *license.key* or *license.bin*.

> ***Example***
> Example of a (invalid) license key in a textual representation:
>
> KgvYGohPOXOGeqY4vppj49wg7PSYg4syn0h9GPFPIlQd9jilxha1mxs81f9QvNUg

System Parameters

The system parameters are a key specific to the implementation of the application required to the generation of an unique license key. They may be obtained from the graphical user interface or through the web-service /about directly from the application.

License key installation

The installation of the license key may also depends on the application, but here are the three modes of installation:

1. Most of the type the license key will be **installed through the application graphical interface** in a dedicated place relative to the administration of the application.

2. You can also **install the license key manually** into the application with the following procedure:

   - Shut down the application.
   - Identify the configuration folder under the application home directory.
   - In that folder, open the *osgi.cm.ini* configuration file. Find or create the *[Runtime]* section in that file.
   - In that file, type the text *key=* followed by the actual license key, on one line. If you have a key-file, replace the value with the path to this file (note that in that case the key-file is not deleted).
   - Save the *osgi.cm.ini* file and restart the application.

3. If you have a **key-file**, named *license.key* or *license.bin*, you can also install this key with the following procedure:

   - Shut down the application.
   - Copy this file into the application home directory.
   - Start the application, once started the application should load this file and delete it.

If available, **the usage of the graphical interface must be the preferred way to install the license key**, as this method will give you a direct feed back of the installation. For all three modes of installation, the log files of the application will contain the result of the installation of the license key.

16 Database installation and upgrade

Chapter Summary

16.1 Database installation

By default, the product is packaged with a pre-built **H2 database**. A dump of a PostgreSQL database is also provided to allow to install the database on a PostgreSQL server.

16.1.1 H2 database

H2 is an embedded database, written in Java. It allows a direct access to the product database with a good performance. In this configuration the database is, by default, stored in the same folder than the server program, and it has access to the data without the requirement of a database server. This ensures that the data are not exposed.

The way the product's server is preconfigured, no configuration modification is required to work with this database. You can start the server immediately.

16.1.2 PostgreSQL

The product's server is delivered with an SQL dump of the database for PostgreSQL. The usage of this database requires a PostgreSQL server installed on your infrastructure, on the same host or any accessible other host.

To install this database, follow the subsequent steps before the first start of the server:

Step 1 Do not start the server.

If the installation wizard allows you to automatically start the server, deselect this option. If the server starts anyway, turn it down with the normal procedure before starting this installation process. In this last case, ignore any error that may occur when starting the server, as the server is not made to be able to start without its database.

The important point here is that the whole procedure must be done with **the server turned down**.

Step 2 Import the PostgreSQL database's SQL dump.

Locate the SQL dump file, in the sub-folder: **./database/migrate** of the application's home directory. The SQL dump file name is **postgresql.sql**.

To import the database file, use the **psql** command provided by PostgreSQL.

Type the following command:

```
psql --host=[host] --port=[port] --username=[user] --dbname=[dbName] -f
./database/migrate/postgresql.sql
```

where:

- **[host]** is the PostgreSQL server hostname.
- **[port]** is the PostgreSQL server TCP-IP port number.
- **[user]** is the user login used to connect to the database.
- **[dbname]** is the database name.

Step 3 Reconfigure the server.

To access the new database, the application's server need to be reconfigured.

You need to change the parameter defining your data source type from "h2" to "postgresql". This parameter may be not defined. In that case, it is not required to change it.

You need to change the parameter defining your data source's JDBC URL to use PostgreSQL and define the login and the password associated with the PostgreSQL database.

> ***Reference***
> Refer to the *Configuration Guide* of the server to know where to find this configuration and how the update it.

16.2 Database backup

Keep a backup of the application's database, periodically or before any upgrade operation, is an important step.

To generate a backup of the H2 Database, you may use the following command, assuming that it is run from the application server home directory:

```
java -cp ./plugins/h2-1.4.199.jar org.h2.tools.Script -url [url] -user [user] -
password [password] -script ./database/backup/H2_[Date].sql compression zip cipher
AES password [zip_password]
```

where:

- **[url]**, **[user]** and **[password]** are the corresponding parameter used in the server configuration file to access to the H2 database.
- **[date]** is any time stamp that should be used to identify the backup file.
- **[zip_password]** is a second password used to encrypt the backup file.

> **Note**
> Depending on the H2 database connection, the application server may need to be running during this operation. This is required if the URL start with *jdbc:h2:tcp:*.

> ***Reference***
> To generate a backup of the PostgreSQL database, refer to the PostgreSQL documentation (Chapter 26, Backup and Restore).

16.3 Database upgrade

The database upgrade is required before the first start of the new version of the application server. To proceed to this upgrade you have two options:

- to upgrade manually, which requires a direct connection to the database and the execution of SQL scripts, or
- to upgrade automatically, using a command line program.

The whole operation may require some minutes according to the actual size of the database and the number of versions to upgrade to the current version.

16.3.1 Prerequisites

The following steps must be executed before to upgrade the database.

1. The application's server must be updated to the latest version and **not** started.
2. A backup of the application's database is highly recommended. Note that the command line program may generate an automatic backup of the H2 database if you use one. In that case, free disc space is required in the **./database/backup/** folder.

16.3.2 Manual upgrade

To manually upgrade your database, execute the appropriate SQL scripts stored in the server's installation sub-folder: **./database/upgrade/**. Depending on number of versions to upgrade.

Step 1 Connect to the database.

First of all you have to connect to your database, through any SQL tool you have to your disposition.

For an H2 database, you may use the H2 Console command line program to open a temporary H2 web-console.

> ***Warning!***
> This web-console must not stay opened as it may expose your database server to serious threats.

Step 2 Locate the Database Version Table.

In the application's database schema, locate the ARCADDBV table and check that this table contain the following columns:

- DBV_CODE a varchar column used to identify the database component.
- DBV_VERSION an integer column used to store the database component version number.
- DBV_UPDATE a date column storing the actual date of the installation of the corresponding database's version.

If some of these columns do not exist or if the ARCADDBV table does not even exist, you will have to execute the **./database/upgrade/initialize.sql** script file. If this file does not exists, contact the ARCAD software support line to get the special upgrade operation required to your case.

Open this file in a text editor and execute the SQL orders, or load it directly with your SQL console.

Step 3 Check the current AFS component version.

Check the highest version number of the "AFS" component code in the ARCADDBV table. You can do it with the following command:

```
SELECT MAX(DBV_VERSION) FROM ARCADDBV WHERE (DBV_CODE = 'AFS')
```

This number is the current AFS database version number. The "AFS" part of the database is shared across different ARCAD applications' servers. Depending on your application, you may not have any component installed related to AFS. Therefore, if the request above does not return any results, go to the step **6**.

Step 4 Check if there is an upgrade file for this version.

In the folder **./database/upgrade/** , look for the SQL scripts file to use for the ugrade. This file is named "AFS_V*X*.sql" and/or "AFS_V*X*_*DB*.sql" where *X* is this new version number and *DB* is the database identifier corresponding to your database:

- _h2 for H2 Database
- _pg for PostgreSQL database.

You need to select the file with a version number directly higher than your current version. For instance, if the current AFS component database is 3, then select file named "AFS_V4*.sql". If your database is stored under PostgreSQL,look for the file named "AFS_V4_pg.sql", or if your database is stored into H2 Database execute the "AFS_V4_h2.sql" file.

If there are no files corresponding to targeted version number, then the AFS database component is up to date. This part of the process is completed to can go to the step number **6**.

Step 5 Execute the SQL script files.

The file "AFS_V*X*.sql" contains the generic part of the SQL code. If it exists, it must be executed first. Then the "AFS_V4_*DB*.sql" files contain the database specific code. Execute the one which corresponds to the type of your target database.

Open the identified new version files in a text editor and execute the SQL orders, or load it directly with your SQL console.

Once the script executed, return back to the step number **3** to check if another upgrade is required.

Step 6 Check the current Application database version.

Once the AFS database part is updated, you have to do the same operation for the application related part of the database. This component is called "PRODUCT" in the ARCADDDBV table.

To do so, execute the following SQL command to find the current version of the application database:

```
SELECT MAX(DBV_VERSION) FROM ARCADDBV WHERE (DBV_CODE = 'PRODUCT')
```

This will return the current database version number.

Step 7 Check if there is an upgrade file for this version.

In the **./database/upgrade/** folder, look for the SQL files corresponding to the next version number.

This file will be named "PRODUCT_V*XXX*.sql" and/or "PRODUCT_V*XXX*_*db*.sql" where *XXX* is the next number version and _*db* is either _h2 or _pg.

If there are no files corresponding to targeted version number then the database upgrade process is completed.

Step 8 Execute the SQL file script.

Again, if there is a generic SQL file "PRODUCT_V*XXX*.sql" execute it first, and after this one execute the database specific "PRODUCT_V*XXX*_*db*.sql" corresponding to your database target. To execute the SQL files, load them directly from the SQL console or by copy/paste from a text editor, and execute the SQL orders contained into them.

And once the scripts are executed, return to the step number **6** to check if another upgrade is required.

16.3.3 Automatic upgrade

The automatic update command requires that the application server is stopped and it must be executed immediately after an update before the first start of this server. This command must be executed from a shell, with a user with the privileges required to access the files of the application server. This may require an elevation. It will performs the following actions:

- Retrieve the database access parameter from the application configuration. Then the execution of this program does not require any specific parameter.
- If the target database is an H2 Database, makes a backup of the database, and regenerates the database from this backup to clean up the internal structure and indexes of it. The database backup is stored into the folder **./database/backup**. This backup file is compressed but you must ensure that the disc has enough free space to store this backup.

- If the target is not an H2 database, the program prompts the user to manually perform a backup of the database. To disable this prompt, use the option **-noprompt**, see below.
- If the target database is an empty database, and the option **-install** is used, it creates the database content before updating it to the latest version level.
- If the database exist but is too old to be processed, tries to update it with the ./database/upgrade/initialize.sql script. Note that the product update process may require to install a certain version of it before installing the current version.
- Executes the same process of upgrade, applying all versions of the SQL script in order to obtain the higher version level available.

To run this command, use the following command call, from the **./tools** folder of the server's installation home directory:

Linux/Unix: `./dbupdate.sh`

Windows: `./dbupdate.bat`

This command accept the following optional parameters:

`-i, -install`
: This option forces the database creation if the current database target is an empty database.

`-np, -noprompt`
: This option disables the user prompt asking for a manual backup of a non H2 Database.

`-p <password>, -password <password>`
: This option defines the password to use for the H2 Database backup file. If not defined, the password used is the one used to connect to the database. If a blank password is given, then no password i used to encrypt the backup file, which is not recommended.

`-ncdb, -nocleandb`
: If used, this option disables the "clean up" process of the H2 database. Note that this process may restore the performances of the H2 database and test the validity of the backup archive, but it may also corrupt the database if an error occurs.

16.4 Database migration

The database migration allows to move the application data from an H2 database to a PostgreSQL database.

There are two ways to migrate the database. You can choose to do an automatic or a manual migration.

16.4.1 Planning your migration

- The migration is a one way process. Once migrated to PostgreSQL, there is no migration process to go back to an H2 database.
- The migration is an offline process. During the whole process, the application server will be not accessible. The migration process is rather quick, though.

- The migration is done to a same version level. It requires that the original H2 database is upgraded first.

16.4.2 Prerequisites

The following step must be executed before to migrate the database.

1. Turn off the application's server.
2. Create a backup of the H2 database.
3. If necessary, upgrade the H2 database to the current version level, see Database upgrade.
4. Create a user for the application's server and install the PostgreSQL database from the provided current version dump. The whole migration process must be done with the application's user.

16.4.3 Migration scripts

To perform the migration, you need to use the following scripts. These scripts are stored into the **./database/migrate/** folder.

Script	Purpose
./database/migrate/create_fk_constraints.sql	Creation of referential integrity constraints
./database/migrate/drop_fk_constraints.sql	Drop of referential integrity constraints
./database/migrate/export_H2_csv.sql	H2 commands to export table data to csv
./database/migrate/import_postgresql_csv.sql	PostgreSQL command to import csv to table
./database/migrate/set_sequences.sql	Set sequences number.

16.4.4 Manual migration

Scripts on PostgreSQL are executed with its tool **psql**. This command must installed on the host where the migration is executed.

The connection parameters are entered either as parameters and values or as JDBC URL.

Run the following steps in order, to migrate data from H2 to PostgreSQL with the help of the scripts provided in the last chapter.

Step 1 Export of H2 data to csv files.

Execute the sql file **export_H2_csv.sql** on H2 database with the H2 **RunScript** tool.

The csv files are created in the user home directory.

The script can be executed again as it replaces the existing csv files.

```
java -cp plugins/h2-1.4.199.jar org.h2.tools.RunScript -url [H2 jdbc url] -
```

```
user [user] -password [password] -script export_H2_csv.sql -showResults
```

The ***-showResults*** outputs the following line for each call in the console which shows the number of lines exported for a table.

CALL CSVWRITE('~\str_gitexport_properties.csv', 'SELECT gip_id , gip_git_id , gip_prp_id , gip_upddate , gip_deleted FROM str_gitexport_properties','charset=UTF-8');

--> 1

Step 2 Drop of integrity referential constraints.

Execute the sql file **drop_fk_constraints.sql** on PostgreSQL with the command ***psql***.

If there is an error, no constraints are dropped; then the script can be executed again.

```
psql --host=[host] --port=[port] --username=[user] --dbname=[dbName] --
single-transaction -f drop_fk_constraints.sql 1> drop_fk_constraints.log 2>
drop_fk_constraints.err
```

In the example above, the SQL file, the execution log file ***drop_fk_constraints*.log** and the errors file ***drop_fk_constraints*.err** are in the working directory.

If the user password is not set in the PostgreSQL password file, the password is asked at the prompt.

Step 3 Data import from csv files to PostgreSQL tables.

Execute the SQL file **import_postgresql_csv.sql** on PostgreSQL with the command ***psql***.

Before loading the data from the csv file, the table is truncated. In case of any errors the script can be executed again without duplicates.

> ***Warning!***
> Execute the psql command from the directory where csv files are stored.

```
psql --host=[host] --port=[port] --username=[user] --dbname=[dbName] -f
import_postgresql_csv.sql 1> import_postgresql_csv.log 2> import_postgresql_
csv.err
```

In the example above, the SQL file, the execution log file **import_postgresql_csv.log** and the errors file **import_postgresql_csv.err** are in the working directory.

If the user password is not set in the PostgreSQL password file, the password is asked at the prompt.

Step 4 Set of sequences number.

Execute the SQL file **set_sequences.sql** on PostgreSQL with the command ***psql***.

The script can be executed again in case of any errors.

```
psql --host=[host] --port=[port] --username=[user] --dbname=[dbName] -f set_
sequences.sql 1> set_sequences.log 2> set_sequences.err
```

In the example above, the SQL file, the execution log file **set_sequences.log** and the errors file **set_sequences.err** are in the working directory.

If the user password is not set in the PostgreSQL password file, the password is asked at the prompt.

Step 5 Creation of integrity referential constraints.

Execute the SQL file **create_fk_constraints.sql** on PostgreSQL with the command ***psql***.

If there is an error, no constraints are created; then the script can be executed again.

```
psql --host=[host] --port=[port] --username=[user] --dbname=[dbName] --
single-transaction -f create_fk_constraints.sql 1> create_fk_constraints.log
2> create_fk_constraints.err
```

In the example above, the SQL file, the execution log file ***create_fk_constraints*.log** and the errors file ***create_fk_constraints*.err** are in the working directory.

If the user password is not set in the PostgreSQL password file, the password is asked at the prompt.

Step 6 Change the configuration of the application's server.

Refer to the documentation of the server to know how to update the server's configuration. Replace the H2 JDBC URL, the login and the password with the new PostgreSQL ones.

Step 7 Restart the application's server.

> ***Reference***
> Refer to the application's server documentation, in the section concerning the installation for your platform.

16.4.5 Automatic migration

To perform an automatic migration from H2 to PostgreSQL, execute the following command from the **./tools** folder in a system shell, with a user owning the required privileges.

> **Important!**
> An elevation of privileges may be required depending of your system.

Execute the following command:

Linux/Unix: `./dbmigration.sh`

Windows: `./dbmigration.bat`

You will be prompted for the required parameters. To avoid this, you can pass the following parameters to this command:

`<url>`

The JDBC URL used to connect to the target PostgreSQL database. This URL must look like: jdbc:postgresql://<host>[:<port>]/]<database> where *host* is the PostgreSQL server host name, *port* is the TCP port number if different from the default one, 5432, and *database* is the PostgreSQL database name. Other option may be added to this URL, refer to the official PostgreSQL JDBC documentation.

`-l <user>, -login <user>`

The user login used to connect to the PostgreSQL database. Note that this user must be associated with a default schema where the application database has been created.

`-p <password>, -password <password>`

The corresponding user password.

`-nb, -nobackup`

If used, this parameter disables the backup of the source database before to release to the migration.

If any of the connection's parameters are missing, it will be prompted in the command shell.

This command proceeds to the whole database migration, according to the previous chapter, and update the configuration of the application's server.