InfoFabrik gmbh

ReportServer

Configuration Guide 5.0

InfoFabrik gmbh

ReportServer

Configuration Guide 5.0

InfoFabrik GmbH, 2025

http://www.infofabrik.de/
http://www.reportserver.net/

Contents

Contents i

1 Preamble 3

2 Installation 5
2.1 Automatic Installation . . . 5
2.2 Manual Installation . . . 6
2.3 Running ReportServer on JBoss Wildfly . . . 8
2.4 Installing the Demo Data . . . 9

3 External Configuration Files 11
3.1 `persistence.properties` . . . 11
3.2 `reportserver.properties` . . . 13

4 Configuration 19
4.1 Datasources . . . 20
4.2 Datasinks . . . 22
4.3 Dynamic Lists . . . 24
4.4 Setting up the Scheduler . . . 24
4.5 Remote RS Server Settings . . . 33
4.6 Transport settings . . . 34
4.7 Report execution error log settings . . . 36
4.8 Export settings . . . 40
4.9 UI Customization . . . 41
4.10 Extensions . . . 54
4.11 Executing Reports using URLs . . . 54
4.12 Misc Settings . . . 55
4.13 Scheduler Settings . . . 55
4.14 Localization Settings . . . 56
4.15 Security related properties . . . 58
4.16 SSO related settings . . . 63
4.17 TeamSpace related settings . . . 68

5 External Configdir **71**

A Config File Reference **75**

Chapter 1

Preamble

Business Intelligence

Business Intelligence (BI) describes the ability to jointly analyze all of a company's data, distilling relevant information to be used to foster better business decisions. The foundation of any BI solution is the careful preprocessing of existing data, for example, in a data warehouse.

ReportServer acts as the gateway between end-users and the collected data, allowing users to efficiently access and analyze the available data. From camera-ready evaluations to fine-grained ad-hoc reporting; ReportServer provides you with the tools to support your daily work.

Target Audience

This document is designed for future administrators of ReportServer.

Separate manuals and instructions illustrate the various aspects of ReportServer.

ReportServer Configuration Guide: Describes the installation of ReportServer as well as the basic configuration options.

ReportServer User Guide: The user guide describes ReportServer from the point of view of the ultimate user. It includes an in-depth coverage of dynamic lists (ReportServer's adhoc reporting solution), execution of reports, scheduling of reports, and much more.

ReportServer Administrator Guide: The administrator guide describes ReportServer from the point of view of administrators that are tasked with maintaining the daily operation of the reporting platform including the development of reports, managing users and permissions, monitoring the system state, and much more.

ReportServer Scripting Guide: The ReportServer scripting guide covers the scripting capabilities of ReportServer which can be used for building complex reports as well as for extending the functionality of ReportServer or performing critical maintenance tasks. It extends the introduction to these topics given in the administrator guide.

Chapter 2

Installation

ReportServer is a web-application based on the Java Servlet technology and, thus, runs in an application server (such as Apache Tomcat). Being a Java application ReportServer supports any operating system that has a Java runtime environment and for which a supported application server is available. All application metadata is stored in a relational database.

ReportServer is available for download in .zip file format for deployment within an application server. Additional options, including native installers for Windows, docker images are available here: http://reportserver.net/download.

While the manual installation provides the most flexibility, it requires some prior knowledge about the deployment of web applications and the operating of an application server, so we recommend this for system administrators and advanced users. The Windows packages on the other hand are completely self contained and can be installed with only a couple of clicks.

You can download ReportServer from http://reportserver.net/download.

2.1 Automatic Installation

The ReportServer installer stack is available for Windows and provides a pre-configured installation of ReportServer with Apache Tomcat as application server and MariaDB as database backend. The installers are available from http://reportserver.net/download. The installation wizard guides you through the installation providing you with the option to pre-configure parts of ReportServer and you can choose whether to install the demo package or not (we note that you can also manually install the demo data later; see Section 2.4). In case you already have an application server or MariaDB database running and the default ports are in use, you are asked to provide alternative ports.

Installation on Linux machines should be done manually or using the docker images available here: http://reportserver.net/download.

2.2 Manual Installation

The manual installation allows you to fine-tune the installation process to your environment and is generally recommended for any production environment.

Installation of the Java Runtime Environment (JRE)

ReportServer requires an installed Java Runtime Environment (JRE) in version 11 or newer.

Further, you need some extra configuration which can be done in the setenv.bat / setenv.sh of your Tomcat environment. Specifically, the following configuration is needed:

- -add-opens=java.base/java.net=ALL-UNNAMED
- -add-opens=java.base/jdk.internal.ref=ALL-UNNAMED
- -add-opens=java.base/jdk.internal.reflect=ALL-UNNAMED
- -add-opens=java.base/java.lang.invoke=ALL-UNNAMED
- -add-opens=java.base/java.util=ALL-UNNAMED
- -add-opens=java.base/java.lang.ref=ALL-UNNAMED
- -add-opens=java.base/java.lang.reflect=ALL-UNNAMED
- -add-opens=java.base/sun.reflect.generics.repository=ALL-UNNAMED
- -Djavax.net.ssl.trustStoreType=JKS
- -Dfile.encoding=UTF-8
- -Djava.awt.headless=true

The following is an example configuration in our Windows packages where the configuration mentioned above is integrated:

```
set
JAVA_OPTS=++JvmOptions="-Drs.configdir=C:/infofabrik/↙
    ↳ reportserverenterprise-x.x.x-yyyy-0/apps/reportserver/↙
    ↳ reportserver-conf"
++JvmOptions="-Dfile.encoding=UTF-8" ++JvmOptions="-Djava.awt.headless↙
    ↳ =true" ++JvmOptions="--add-opens=java.base/java.net=ALL-UNNAMED" ↙
    ↳ ++JvmOptions="--add-opens=java.base/jdk.internal.ref=ALL-UNNAMED"↙
    ↳  ++JvmOptions="--add-opens=java.base/jdk.internal.reflect=ALL-↙
    ↳ UNNAMED" ++JvmOptions="--add-opens=java.base/java.lang.invoke=ALL↙
    ↳ -UNNAMED" ++JvmOptions="--add-opens=java.base/java.util=ALL-↙
    ↳ UNNAMED" ++JvmOptions="--add-opens=java.base/java.lang.ref=ALL-↙
    ↳ UNNAMED" ++JvmOptions="--add-opens=java.base/java.lang.reflect=↙
    ↳ ALL-UNNAMED" ++JvmOptions="--add-opens=java.base/sun.reflect.↙
    ↳ generics.repository=ALL-UNNAMED" ++JvmOptions="-Djavax.net.ssl.↙
    ↳ trustStoreType=JKS" --JvmMs 512 --JvmMx 1536 set JDK_JAVA_OPTIONS↙
    ↳ = %JDK_JAVA_OPTIONS%
```

Tip. If the host computer supports it, you should use the 64-bit edition.

Installation and configuration of the application server

ReportServer can be configured to run in any application server that supports the Java Servlet Technology (e.g., Jetty, Tomcat or JBoss Wildfly). We recommend using Apache Tomcat (http://tomcat.apache.org/).

In order to smoothly run ReportServer it is necessary to provide the application server with sufficient memory which usually means that you have to increase the default values. If too little memory is available then ReportServer might not start at all or your users might experience performance problems. The following recommendations are to be understood as lower bounds. Depending on your environment (the types of reports that you want to run and the number of users the system is to handle) you might need to increase these (especially the available heap size).

We recommend to set the available permanent generation space (PermGenSpace) to at least 256mb (better 512mb). The maximal available heap size should be at least 1.5gb.

Furthermore, the encoding should be set to UTF8.

A sample configuration of the VM might look as follows:

```
-Xmx4096M
-XX:MaxPermSize=512M
-Dfile.encoding=UTF8
```

Further information can be found, for example, at http://www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html.

Unpack the zip-archive

Stop the application server, if it was running, and unpack the ReportServer archive to a directory called reportserver below the webapps directory of your application server (on Windows this could be, for example, `C:\Program Files\Apache Tomcat\webapps`; on `Linux /var/tomcat/webapps`).

Setup the database

Before we can start the application server we need to configure the database connection that ReportServer uses to store its metadata. ReportServer internally uses JPA with Hibernate (http://www.hibernate.org/) which allows us to support most popular database systems. A list of the database systems supported by Hibernate can be found at https://developer.jboss.org/docs/DOC-13921.

Installation of the JDBC driver

ReportServer comes bundled with drivers for the open source databases MySQL (http://www.mysql.com) and PostgreSQL (http://www.postgresql.org). If you use either of these databases you do not need to manually copy the jdbc driver to the `lib` directory. If you use any other database

you need to copy the corresponding JDBC driver to ReportServer's `lib` directory, that is, to directory:

`path_to_webapps/reportserver/WEB-INF/lib`.

Creating the ReportServer schema

Next we need to setup the necessary database tables. For this, choose the create script corresponding to your database system from the directory "ddl" (directly beneath the `reportserver` directory) and execute it on your database. For a PostgreSQL database, for example, use the file:

`reportserver-RSx.x.x-yyyy-schema-PostgreSQL_CREATE.sql`.

Adapt the `persistence.properties` config file

To complete the database setup we need to configure the connection properties. For this, go to `WEB-INF/classes` and edit the file `persistence.properties`. The file `persistence.properties` contains the configuration of the ReportServer database connection. You can find further information on this config file in Chapter 3. Following is a sample configuration for MySQL.

```
hibernate.dialect=net.datenwerke.rs.utils.hibernate.MySQL5Dialect
hibernate.connection.driver_class=com.mysql.cj.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost:3306/reportserver
hibernate.connection.username=root
hibernate.connection.password=root
```

The config file `reportserver.properties`

The final step in the installation is to check the (and possibly adapt) the main configuration settings. These are stored in the the config file `reportserver.properties` which is located in the `WEB-INF/classes` directory. It contains basic properties concerning the available authentication procedures used by ReportServer as well as configuration for cryptopgraphic functionality used in ReportServer.

A detailed description of the available parameters are given in Chapter 3 External Configuration Files.

Application Server Start

You can now start the application server. Once the application server is started up ReportServer should be accessible, for example, under the URL `http://localhost:8080/reportserver`. ReportServer has generated the root user with which you can login:

```
username: root
password: root
```

2.3 Running ReportServer on JBoss Wildfly

To run ReportServer on JBoss Wildfly only a few configuration options need to be considered. First, you should ensure that JBoss is configured to use sufficient memory. The necessary changes can be made in file wildfly/bin/standalone.conf (or standalone.conf.bat for windows systems). To run

ReportServer you should provide JBoss with at least 1.5 GB of heap space. Depending on the number of users, this value should be increased. Following is an example configuration:

```
JAVA_OPTS="-Xms64m -Xmx2g -XX:MaxPermSize=256m -Djava.net.↙
    ↳ preferIPv4Stack=true"
```

Besides increasing the memory as described above, you will need to add an application descriptor file called `jboss-deployment-structure.xml`. You will need to place this configuration file into the WEB-INF directory of ReportServer. Following is the content of the descriptor.

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-deployment-structure>
  <deployment>
    <exclusions>
      <module name="org.hibernate" />
      <module name="org.antlr" />
    </exclusions>
    <exclude-subsystems>
      <subsystem name="weld" />
      <subsystem name="org.hibernate" />
      <subsystem name="org.hibernate.validator" />
      <subsystem name="org.antlr" />
      <subsystem name="jpa" />
    </exclude-subsystems>
    <dependencies>
      <module name="org.bouncycastle" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Having increased the memory and added the jboss-deployment-structure.xml to the ReportServer WEB-INF directory, you are now good to go.

2.4 Installing the Demo Data

ReportServer comes with a demo data package containing the example setup of the fictionally toy company "1-to-87". In order to install the demodata package you should login with a root account (if you used the Windows installer, this would be the account you setup during installation). Also note that the **installation of the demodata will remove any existing data in the system**. To install the demo package login to ReportServer and open the terminal by pressing

```
CTRL+ALT+T
```

Then, to initiate the installation, run the command

```
pkg install -d demobuilder-VERSION_NR
```

Here, "VERSION_NR" must be replaced by the correct version, which can be obtained by using the autocomplete feature of the terminal. Simply hit `TAB` to get a list of options:

```
pkg install -d [TAB]
```

Chapter 3

External Configuration Files

ReportServer has only two (external) config files which hold information on the database connection as well as information on available authentication methods. All other configuration is done from within ReportServer. The file `persistence.properties` (in directory `WEB-INF/classes`) holds information on the ***database connection*** that is used by ReportServer. The configuration file `reportserver.properties` (in the same directory) holds information about ***available authentication schemes***, as well as, ***cryptography related properties***.

By default the external configuration files are located in `WEB-INF/classes` and thus within the web-apps folder. It is advisable to move these files to an external location as this allows easier upgrades to future versions. For a detailed description of how to use an external configuration dir see Chapter 5.

3.1 `persistence.properties`

ReportServer uses the Java Persistence API (JPA) to abstract from the actual database system when storing application data. The necessary configuration is made in the `persistence.properties` config file.

Example

```
hibernate.dialect=net.datenwerke.rs.utils.hibernate.MySQL5Dialect
hibernate.connection.driver_class=com.mysql.cj.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost:3306/reportserver
hibernate.connection.username=rs
hibernate.connection.password=rs
```

Note that to configure the JPA/Hibernate settings, we are not editing the standard JPA configuration file (usually called persistence.xml) but a ReportServer properties file.

Connection properties

ReportServer supports all databases that are supported by Hibernate. A list of the database systems supported by Hibernate can be found on the hibernate webpages[1]. We recommend to run

[1] https://developer.jboss.org/docs/DOC-13921

ReportServer on one of the following databases for which we now give example configurations:

Example config for *MySQL*

```
# MySQL
hibernate.dialect=net.datenwerke.rs.utils.hibernate.MySQL5Dialect
hibernate.connection.driver_class=com.mysql.cj.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost:3306/reportserver
hibernate.connection.username=rs
hibernate.connection.password=rs
```

Note the custom dialect `net.datenwerke.rs.utils.hibernate.MySQL5Dialect`.

Example config for *MariaDB*

```
# MariaDB
hibernate.dialect=net.datenwerke.rs.utils.hibernate.MariaDbDialect
hibernate.connection.driver_class=org.mariadb.jdbc.Driver
hibernate.connection.url=jdbc:mariadb://localhost:3306/reportserver
hibernate.connection.username=rs
hibernate.connection.password=rs
```

Example config for *PostgreSQL*

```
# PostgreSQL
hibernate.dialect=net.datenwerke.rs.utils.hibernate.↙
    ↳ PostgreSQLDialect
hibernate.connection.driver_class=org.postgresql.Driver
hibernate.connection.url=jdbc:postgresql://localhost/postgres
hibernate.connection.username=rs
hibernate.connection.password=rs
hibernate.connection.autocommit=false
```

Example config for *Oracle*

```
# Oracle
#
# Select ONE of the following dialects depending on your Oracle ↙
    ↳ version
#
hibernate.dialect=net.datenwerke.rs.utils.hibernate.Oracle10gDialect
#   hibernate.dialect=net.datenwerke.rs.utils.hibernate.↙
    ↳ Oracle12cDialect
#
hibernate.connection.driver_class=oracle.jdbc.driver.OracleDriver
hibernate.connection.url=jdbc:oracle:thin:@localhost:1521:MYDB
hibernate.connection.username=rs
hibernate.connection.password=rs
hibernate.connection.autocommit=false
```

Example config for *SQL Server*

```
# SQL Server
hibernate.dialect=net.datenwerke.rs.utils.hibernate.↙
    ↳ SQLServer2008Dialect
```

```
hibernate.connection.driver_class=com.microsoft.sqlserver.jdbc.↙
    ↳ SQLServerDriver
hibernate.connection.url=jdbc:sqlserver://localhost/sqlserver:1433;↙
    ↳ databaseName=mydb
hibernate.connection.username=rs
hibernate.connection.password=rs
hibernate.connection.autocommit=false
```

Note, that the JDBC driver corresponding to your database must be copied to the directory

`path_to_webapps/reportserver/WEB-INF/lib` or to your external-configuration directory.

Connection Pool (C3P0) Settings

Hibernate uses the C3P0 connection pool. The following properties allow to configure C3P0 as used by Hibernate. Note that this does not have any effect on the connection pool used by ReportServer for handling reporting.

```
  hibernate.c3p0.acquire_increment=5
hibernate.c3p0.idle_test_period=60
hibernate.c3p0.timeout=3600
hibernate.c3p0.max_size=30
hibernate.c3p0.max_statements=0
hibernate.c3p0.min_size=5
```

See also http://www.mchange.com/projects/c3p0/index.html#configuration.

The most commonly used properties are:

Property	Description
`acquire_increment`	Defines how many connections are acquired simultaniously by c3p0, if all connections currently in the pool are busy.
`idle_test_period`	If not zero, C3P0 will test idle connections in this intervall.
`timeout`	Number of seconds a pooled connection can remain idle before it is discarded. Zero means that idle connections are never discarded.
`max_size`	Maximum number of connections in the pool.
`max_statements`	Maximal size of the statement cache. Zero means that statements should not be cached.
`min_size`	The minimum number of connections to be kept in the pool.

3.2 reportserver.properties

The config file `reportserver.properties` contains settings which are needed at ReportServer startup time as well as settings concerning cryptographic functionality. All properties are stored as attribute value pairs.

Excerpt from the `reportserver.properties` file

```
rs.crypto.pbe.passphrase = The Passphrase
rs.crypto.pbe.keylength = 128
```

Crypto settings

Passwords (such as, for example, datasource passwords[2]) that are stored in ReportServer will be encrypted. ReportServer uses AES and password based encryption. For this, you need to configure the following properties:

– `rs.crypto.pbe.salt`

the salt that is used on key generation by the password based encryption method. This value should be set to a long random string.

– `rs.crypto.pbe.keylength`

The key size used. Keep in mind that key sizes over 128 require the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files. For more information, see http://www.oracle.com/technetwork/java/javase/downloads/index.html.

For secure storage of user passwords ReportServer uses the salted HMAC construction.

– `rs.crypto.passwordhasher.hmac.passphrase`

Defines the static part of the salt for the salted HMAC construction. This value should be set to a long random string.

Authentication settings

ReportServer supports several methods for user authentication. The different methods can also be combined.

– `rs.authenticator.pams`

This parameter defines which authentication methods are to be used. The individual values are separated by a colon.

Note that the default PAMs usually come in two variants, one being called ***Authoritative***, e.g. `UserPasswordPAMAuthoritative`, or `LdapPAMAuthoritative`. The difference between the two variants (the non-authoritative and the authoritative variant) is how they handle the case where they cannot find the necessary information within the provided list of tokens. The authoritative version then denies access, while the non-authoritative version opts for letting someone else decide (can't tell, let another PAM in the list decide). More details on this can be found in the Script Guide.

The following authenticator modules are available

`net.datenwerke.rs.authenticator.service.pam.UserPasswordPAM`

This is the standard mechanism and allows users to authenticate using a username and password. Note: with this authentication method username and password are sent to the server in the clear.

[2]We note that for user passwords only a salted hash is stored. Passwords that need to be recoverable, such as database passwords, are stored encrypted.

Thus, this method should only be used over a secure channel (such as TLS/SSL). If you are not in a secured environment choose the ***ChallengeResponsePAM*** instead.

`net.datenwerke.rs.authenticator.service.pam.IPRestrictionPAM`

With this method you can restrict the set of IP addresses that can access ReportServer. The allowed address ranges are configured in `rs.authenticator.iprestriction.addresses`. The individual values are separated by a colon. Example: `rs.authenticator.iprestriction.addresses` ↵ ↳ `= 127.0.0.1/32:192.168.1.0/24`

`net.datenwerke.rs.authenticator.service.pam.EveryoneIsRootPAM`

This method will disable authentication and log in all users as root.

`net.datenwerke.rs.authenticator.cr.service.pam.ChallengeResponsePAM`

This method is similar to the ***UserPasswordPAM*** but the password is transmitted securely to the server. In cases where the connection is not secured via SSL this is the recommended authentication method. Note though, that this requires the client browser to perform cryptographic tasks which may be slow on non-up-to-date browsers.

`net.datenwerke.rs.authenticator.service.pam.ClientCertificateMatchEmailPAM`

If your organization uses ***x509 certificates***, you can use this method to allow users to log in with their client certificates.

`net.datenwerke.rs.ldap.service.ldap.pam.LdapPAM`

If your organization uses LDAP and you want to allow LDAP authentication, you can use this method to allow users to log in with their LDAP credentials. Note that you previously have to import your LDAP users as described in the Admin Guide: https://reportserver.net/en/guides/admin/chapters/Integrating-ReportServer-with-an-Active-Directory-using-LDAP/

`rs.authenticator.pam.ClientCertificateMatchEmailPAM.debug`

This parameter allows to enable debug mode for client certificate authentication.

– `rs.authenticator.blockroot`

Setting this to true disables direct access using the root user. This is recommended for production environments. Note that you can switch to the root user using sudo if you need to perform administrative tasks (and you have the corresponding privileges). Further information on sudo can be found in the ***administration guide***.

General settings

`rs.install.basedata`

Setting this to `true` installs base data if the database is empty. This only happens during a first

run. This includes the audit log report and other data. Note that this does not mean that the DDLs will be installed automatically. You still have to install the DDLs manually as specified in the Installation Guide.

`rs.scripting.disable`

If this property is set to `true`, scripting is completely disabled, so no scripts are run, including the onlogin and onstartup scripts. Thus, should you find yourself locked out of ReportServer, you can disable any scripts via this property. Since no scripts will be executed you can then login correctly to ReportServer.

`rs.scheduler.disable`

You can completely disable the scheduler by setting this property to `true`. If the property is set to `true` both in `reportserver.properties` and the analogous property is also set to `true` in `/fileserver/etc/scheduler/scheduler.cf`, the property set in `reportserver.properties` is taken into account, while the one set in `/fileserver/etc/scheduler/scheduler.cf` is ignored. This allows you to completely avoid running scheduled jobs if you don't have the possibility to log in and disable the scheduler quickly enough before any jobs are being executed.

Chapter 4

Configuration

ReportServer is now up and running. In the following section we describe configuration options affecting the operation of ReportServer. The present document should be considered as a reference containing a brief description of the various configuration options. Keep in mind that the settings described here affect all areas of ReportServer which to describe is beyond the scope of this ***config guide***. See the ***administrator's*** and ***user's guide*** for further information to the various areas of ReportServer. Sample configurations can be found in Appendix A.

All configuration files described in this section can be found in ReportServer's internal filesystem. You can access the internal filesystem using the administration module (administration/ file system) or using the terminal: you can open the terminal by pressing `CTRL+ALT+T`.

Tip. Default configuration files are created on first run of ReportServer. Later, when upgrading ReportServer to a newer version, it is probable that newly added configuration files will be missing (i.e. all configuration files added between the version originally installed and the version upgraded to). You can use the "diffconfigfiles" command for getting help on this, check the Administration Guide for details. Note that default values are used for configuration files that are not found.

By using the command `editTextFile` you can edit files directly from the terminal. You can also create new files using the command `createTextFile`. For further information on the workings of the terminal see the ***administrator's guide***. Using the graphical user interface, you can select files similar to selecting files when working in the Explorer in your operating system. To do this, go to administration/file system. To edit a file, choose the tab `Edit file`. Note that the edit file tab is only available for text files and if a proper mime-type is specified.

Please note, that after changing a config file you need to run the terminal command `config reload` for the change to take effect.

Configuration files in ReportServer are usually defined in an XML format.

4.1 Datasources

In this section we will cover:

- how to define the default datasource,
- how to define datasource bundles (only available in ReportServer Enterprise),
- how to configure the connection pool,
- how to configure the internal db

Defining the Default Datasource

ReportServer allows to configure a single default datasource. The default datasource can then be accessed with only a single mouse click when working with reports and parameters. The default datasource is set in the file `/fileserver/etc/datasources/datasources.cf`.

You can use the key of the datasource or use the ID of the datasource for the assignment. Use one of the two variants for the configuration:

```
<defaultDatasourceKey>MY_DATASOURCE<defaultDatasourceKey>
<defaultDatasourceId>12</defaultDatasourceId>
```

Remark. Keep in mind that the datasource key is case sensitive.

Configuring the Connection Pool

By default, ReportServer will pool connections to relational database systems. This increases the stability of the system, since you can define an upper limit of simultaneously open connections and, furthermore, it improves the performance at the same time since database connections are kept open and ready for use.

The connection pools can be configured both globally and per datasource. The configuration is done in the file `/fileserver/etc/datasources/pool.cf`.

To not use connection pools, change the attribute “disable” to “true”: <pool disable="false">.

ReportServer uses the library C3P0 (http://www.mchange.com/projects/c3p0/) to perform connection pooling. To globally set a property, set the property within the <defaultconfig> tags. For data-source-specific settings, use the tag <pool16>, where 16 is the ID of the data source.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <pool>
    <defaultconfig>
      <maxPoolSize>40</maxPoolSize>
      <initialPoolSize>10</initialPoolSize>
      <acquireRetryAttempts>10</acquireRetryAttempts>
      <acquireRetryDelay>500</acquireRetryDelay>
```

```
        <checkoutTimeout>60000</checkoutTimeout>
        <maxConnectionAge>7200</maxConnectionAge>
        <maxIdleTime>3600</maxIdleTime>
      </defaultconfig>
      <pool16>
        <acquireRetryAttempts>20</acquireRetryAttempts>
      </pool16>
    </pool>
  </configuration>
```

The possible configuration settings of C3P0 can be found at http://www.mchange.com/projects/c3p0/#configuration_properties. ReportServer simply passes on any set property to C3P0.

Note that as of ReportServer 4.0.0, the current state of the connections (including total max pool size, busy connections and number of connections) is visualized in the "Connection Pool" System Console.

Internal database

ReportServer uses a database to buffer data coming from non-database datasources such as, for example, CSV datasources. This ***buffer database*** is called the ***internal database***. Per default ReportServer uses the its own database for this and creates tables with the prefix ***rs_tmptbl_***. You can change the database to be used as well as configure certain aspects of the internal database via the configuration file ***datasources/internaldb.cf***. The default configuration file is

```
<configuration>
   <internaldb>
      <droponstartup>true</droponstartup>
      <datasource>REPORTSERVER_DATASOURCE</datasource>
   </internaldb>
</configuration>
```

The ***droponstartup*** tells ReportServer to remove any temporary tables on startup. This should only be turned off for debugging purposes as having old temporary tables still available after startup will cause errors when using the internal DB. Via the ***datasource*** tag you can define the datasource (via its key) to be used as the internal database.

SQL limits and parameter options

The datasource parameter (a parameter that can be used to allow the user to select values from a predefined set) can return a single value or a list of values. If a user selects multiple values (or uses many filters in a dynamic list) this might lead to problems with certain database systems. This is due to the fact that the number of values that can be used in IN clauses is limited for most database systems.

You can specify the maximum number of values that are to be used in an `IN` clause in the file `/fileserver/etc/datasources/sql.cf`. ReportServer will then distribute the selected values over multiple `IN`-clauses.

For this set the following parameter

```
<incondition>
  <maxsize>1000</maxsize>
</incondition>
```

in accordance with the prerequisites of the used database system.

The datasource parameter

Queries that run for a long time have an impact on the performance of ReportServer. You can specify how long ReportServer should wait for the results of a parameter query. A parameter query is executed when a report is opened.

You can specify a timeout for long running parameter queries in the configuration file `/fileserver/etc/datasources/parameter.cf`.

Set the parameter `<querytimeout>60</querytimeout>` to stop queries after 60 seconds.

If you want to use the "post-processing" feature of the datasource parameter enable it using the following lines:

```
<postprocessing>
  <enable>true</enable>
</postprocessing>
```

Tip. In general it is good practice to have parameter queries optimized such that they run very fast such as to provide a good user experience.

The post-processing feature, for example, allows to switch parameter values or perform complex string operations. Learn more about datasource parameter post-processing in the ***parameter chapter*** in the ***administrator's guide***.

4.2 Datasinks

Datasinks can be enabled/disabled in the `/fileserver/etc/datasinks/datasinks.cf` file. Here you have an entry similar as the following for each supported datasink type:

```
<configuration>
   <sftp disabled="false" supportsScheduling="true" />
   <ftp disabled="false" supportsScheduling="true" />
</configuration>
```

The `disabled` option controls if the datasink is overall enabled or disabled. Further, scheduling via datasinks can be enabled or disabled via the `supportsScheduling` setting. Note that you can not enable scheduling if the datasink is overall disabled.

Further, you can select a default datasink per type as shown below for email / SMTP datasinks:

```
<email disabled="false" supportsScheduling="true">
    <defaultDatasinkKey>DEFAULT_EMAIL_DATASINK</defaultDatasinkKey>
</email>
```

Here, your default datasink is the datasink with the key DEFAULT_EMAIL_DATASINK. You can select the default datasink by key:

```
<email disabled="false" supportsScheduling="true">
    <defaultDatasinkKey>DEFAULT_EMAIL_DATASINK</defaultDatasinkKey>
</email>
```

or by id:

```
<email disabled="false" supportsScheduling="true">
    <defaultDatasinkId>14</defaultDatasinkId>
</email>
```

SFTP, FTPS, FTP and SCP Datasinks

> Datasinks are supported as of ReportServer 3.4.0. The legacy FTP configuration in `/fileserver/etc/exportfilemd/storage.cf` is ignored as of version 3.4.0.

For the SFTP/FTPS/SCP datasinks to work, you have to add your SFTP/FTPS/SCP host to your `.ssh/known_hosts` file (https://en.wikibooks.org/wiki/OpenSSH/Client_Configuration_Files#~/.ssh/known_hosts) in order to verify the identity of the remote host, thus protecting against impersonation or man-in-the-middle attacks. Its location be configured in the `/fileserver/etc/security/mis` file as follows.

E.g.: Use `ssh-keyscan IP » known_hosts` in order to add the given IP to your `known_hosts` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <knownHosts>/path/to/your/machine/.ssh/known\_hosts</knownHosts>
</configuration>
```

For manually adding a public key to the `.ssh/known_hosts` file, check here: https://en.wikibooks.org/wiki/OpenSSH/Client_Configuration_Files#Manually_Adding_Public_Keys_to_~/.ssh/known_hosts.

E.g.: Use `ssh-keyscan IP » known_hosts` in order to add the given IP to your `known_hosts` file.

Email SMTP Datasinks

As of ReportServer 4.7.0, the old, deprecated and insecure `mail/mail.cf` configuration file is deleted. Please delete it from your system. Instead, Email SMTP datasinks should be used together with a default email datasink, which can be defined in the `/etc/datasinks/datasinks.cf` configuration file as follows.

```
<email disabled="false" supportsScheduling="true">
    <defaultDatasinkName>Default Email Datasink</defaultDatasinkName>
    <!-- or access via ID -->
    <!-- <defaultDatasinkId>14</defaultDatasinkId> -->
</email>
```

You can use the name of the datasink, or use the ID of the datasink, for the assignment. Use one of the two variants above for the configuration.

Note that you can configure a default datasink per datasink type, so this is not limited to email SMTP datasinks.

4.3 Dynamic Lists

In Dynamic Lists, users can use Computed Columns. Computed columns allow users to extend the list of columns by fields which do not exist in the source but which can be constructed from existing fields. Computed Columns are based on SQL.

In `/fileserver/etc/dynamiclists/computedcolumn.cf` you can specify which SQL functions can be used with Computed Columns.

Add those functions that may be used by your users.

Example:

```
<function>abs</function>
```

Note that SQL functions vary depending on the database system.

We strongly recommend against allowing to use functions that can change data. Further, the database user used for report execution should not have write access.

Theming

In ReportServer Enterprise Edition you can change how the PDF and HTML output of a dynamic list is outlined. Further information on this can be found in the Administration Guide.

4.4 Setting up the Scheduler

With ReportServer you can schedule reports and distribute the result either using mail, directly into a ***TeamSpace***, or sent to any datasink configured, (check Section 4.2 for more details). The next section discusses some important settings.

Mail Server Configuration

Note that the old, deprecated, and insecure /fileserver/etc/mail/mail.cf configuration file is deleted as of ReportServer 4.7.0. Please delete it from your system and use email SMTP datasinks instead together with the default email datasink configuration.

In order for ReportServer to be able to send mails you must specify the mail server settings. Make the following configurations in your Email SMTP datasink.

Setting up the SMTP server. Replace the values host, port, username, and password according to your SMTP server.

```
Host: mail.yourmailserver.com
Port: 25
Username: rs@yourmailserver.com
Password: passwordsecret
SSL: false
TLS enable: false
TLS require: false
```

If you are using SSL or TLS please also specify these values. Next, configure the sender name, email address and forceSender options. If the forceSender option is set to true, the emails will be sent using the given (generic) sender details. If set to false, the specific user sending the email will determine the sender details.

```
Sender: rs@yourmailserver.com
Sender name: ReportServer
Force sender: false
Encryption policy: allow_mixed
```

The encryption policy option controls whether or not mails have to be encrypted or whether it is ok to send mails unencrypted if a user's public key is not specified. Choose between `strict` and `allow_mixed`. Note that if you choose `strict` then mails to users that do not have public key registered with ReportServer will not receive any messages.

Details on setting the default email datasink can be found in Section 4.2.

Scheduler settings

ReportServer comes with a powerful scheduler. ReportServer's scheduler allows you to schedule the execution of reports. The executed report can then either be stored in a folder in a ***TeamSpace*** or sent to any datasource configured. Refer to Section 4.2 Datasinks for more details.

The schedule and report recipients are user provided on scheduling. You can configure the messages that ReportServer will send out on certain events. Each message can be customized to your specifications.

ReportServer will send out the following emails:

- email if a report has been placed into a TeamSpace (fileaction),
- email if a report has been sent to an email SMTP server (fileactionEmailDatasink),
- email if a report has been sent to a table datasink (fileactionTableDatasink),
- email if a report has been sent to a SFTP server (fileactionSftp),
- email if a report has been sent to a FTPS server (fileactionFtps),
- email if a report has been sent to a FTP server (fileactionFtp),

- email if a report has been sent to a Samba - SMB/CIFS server (fileactionSamba),
- email if a report has been sent to a SCP (SSH) server (fileactionScp),
- email if a report has been sent to the local filesystem (fileactionLocalFilesystem),
- email if a report has been sent to a Dropbox datasink (fileactionDropbox),
- email if a report has been sent to a Google Drive datasink (fileactionGoogleDrive),
- email if a report has been sent to an Amazon S3 datasink (fileactionAmazonS3),
- email if a report has been sent to a Box datasink (fileactionBox),
- email if a report has been sent to a Printer datasink (fileactionPrinter),
- email if a report has been sent to a Script datasink (fileactionScriptDatasink),
- email on schedule (notification - scheduled),
- email if a schedule job is revoked (notification - unscheduled),
- email if a scheduled job failed (notification - failed)

The following configurations are done in the file `/fileserver/etc/scheduler/scheduler.cf`. To include information such as "the user who created the schedule entry", "the report's name" etc. in your message you can use a variety of expressions. Substitutions are defined in the ReportServer formula language. You will find further information about the ReportServer formula language in the ***Administrator's***, as well as in the ***User Guide***.

Available Substitutions

Expression	Description
`${job.getName()}`	job's name
`${job.getDescription()}`	job's description
`${job.getId()}`	job's ID
`${report.getName()}`	report's name
`${report.getDescription()}`	report's description
`${report.report.getKey()}`	report's key
`${report.getId()}`	report's ID
`${user.getUsername()}`	username
`${user.getFirstname()}`	first name of user
`${user.getLastname()}`	last name of user
`${user.getEmail()}`	user's email address
`${user.getTitle()}`	user's title
`${user.getId()}`	id of user
`${executor}`	job's executor. You can use the same methods above as with user

${scheduledBy}	job's scheduler. You can use the same methods above as with user
${teamspace.getName()}	name of TeamSpace (only available in fileaction)
${folder.getName()}	name of folder in TeamSpace (only available in fileaction)
${folder}	name of folder in FTP server (only available in fileactionFtp)
${message}	the message that was specified by the user on scheduling
${subject}	the subject that was specified by the user on scheduling
${recipients}	A list of the job recipients. Please check below for the exact configuration (list of users)
${owners}	A list of the job owners. Please check below for the exact configuration (list of users)
${filename}	filename as specified by the user (report is scheduled in teamspace)
${nextDates}	date of next execution
${RS_CURRENT_DATE}	current date
${errMsg}	error message on erroneous execution
${stacktrace}	detailed stacktrace on failed execution

List of users For substitution of a list of users (currently supported: list of job recipients and list of job owners), you can use a fluent API that allows you to configure the output exactly as you need. Available methods for this are:

${withSeparator()}	use a given separator between users. Default is a new line.
${addString(“,”)}	add a String, e.g. a comma
${addBlankspace()}	add a blank space
${addNewline()}	add a new line
${addUsernames()}	add usernames
${addFirstnames()}	add first names
${addLastnames()}	add last names
${addEmails()}	add emails
${addTitles()}	add titles
${addIds()}	user ids
${print()}	create the result string. This method has to be called in the last place.

As mentioned, you can use a fluent API for configuring the output. E.g.,

```
${recipients
      .addFirstnames()
      .addBlankspace()
      .addLastnames()
      .addBlankspace()
```

```
      .addString("(")
      .addUsernames()
      .addString(")")
      .print()
 }
```

will print the following:

Barry Jones (bjones)

Diane Murphy (dmurphy)

Gerard Hernandez (ghernande)

Larry Bott (lbott)

If you want to separate the users by a comma instead of a new line, you can enter use the `withSeparator()` method as follows:

```
${recipients
      .withSeparator(", ")
      .addFirstnames()
      .addBlankspace()
      .addLastnames()
      .addBlankspace()
      .addString("(")
      .addUsernames()
      .addString(")")
      .print()
 }
```

which will print the following data:

Barry Jones (bjones), Diane Murphy (dmurphy), Gerard Hernandez (ghernande), Larry Bott (lbott)

Below you can find some example configurations:

Configuration of email message with attached report (successful execution)

```
<mailaction html="false">
<subject>${subject}</subject>
<text>Text of message: ${message}</text>
<attachment>
  <name>rep-${report.getName()}-${RS_CURRENT_DATE}</name>
</attachment>
</mailaction>
```

Configuration on successful execution of report and storage in TeamSpace

```
<fileaction disabled="false" html="false">
    <subject></subject>
    <text></text>
```

```
</xmlcode>
```

Configuration on successful execution of report sent to email SMTP server

```
<fileactionEmailDatasink disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionEmailDatasink>
```

Configuration on successful execution of report sent to table datasink

```
<fileactionTableDatasink disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionTableDatasink>
```

Configuration on successful execution of report and storage in SFTP server

```
<fileactionSftp disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionSftp>
```

Configuration on successful execution of report and storage in FTPS server

```
<fileactionFtps disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionFtps>
```

Configuration on successful execution of report and storage in FTP server

```
<fileactionFtp disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionFtp>
```

Configuration on successful execution of report and storage in Samba server

```
<fileactionSamba disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionSamba>
```

Configuration on successful execution of report and storage in SCP (SSH) server

```
<fileactionScp disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionScp>
```

Configuration on successful execution of report and storage in the local filesystem

```
<fileactionLocalFilesystem disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionLocalFilesystem>
```

Configuration on successful execution of report and storage in Dropbox

```
<fileactionDropbox disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionDropbox>
```

Configuration on successful execution of report and storage in OneDrive - SharePoint (O365)

```
<fileactionOneDrive disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionOneDrive>
```

Configuration on successful execution of report and storage in Google Drive

```
<fileactionGoogleDrive disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionGoogleDrive>
```

Configuration on successful execution of report and storage in Amazon S3 bucket

```
<fileactionAmazonS3 disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionAmazonS3>
```

Configuration on successful execution of report and storage in Box

```
<fileactionBox disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionBox>
```

Configuration on successful execution of report and storage in Printer datasinks

```
<fileactionPrinter disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionPrinter>
```

Configuration on successful execution of report and storage in Script datasinks

```
<fileactionScriptDatasink disabled="false" html="false">
    <subject></subject>
    <text></text>
</fileactionScriptDatasink>
```

Configuration of notifications on scheduling, unscheduling and execution errors

```
<notification disabled="false" html="false">
<scheduled>
  <subject></subject>
<text></text>
</scheduled>
<unscheduled>
```

```
    <subject></subject>
  <text></text>
  </unscheduled>
  <failed>
    <subject></subject>
  <text></text>
  </failed>
  </notification>
```

If you would like to send emails in the HTML format please set the corresponding html attribute to "true".

In case you do not want to have one or more notifications, you can disable the individual notifications using the disabled attribute:

```
<fileaction disabled="true" html="false">
```

Advanced properties

The subsequent section outlines several advanced scheduler properties, which are configured within the file `/fileserver/etc/scheduler/scheduler.cf` configuration file.

Disabling the scheduler

If you do not want to use the scheduler you can disable it using

```
  <properties>
    <disabled>true</disabled>
  </properties>
```

You can also disable the scheduler by setting the following property in your `reportserver.properties` file:

```
rs.scheduler.disable = true
```

If the property is set both in `reportserver.properties` and in `/fileserver/etc/scheduler/scheduler.cf`, the property set in `reportserver.properties` is taken into account, while the one set in `/fileserver/etc/scheduler/scheduler.cf` is ignored.

Keep in mind that this changes will only take effect after reboot. To enable or disable the scheduler while ReportServer is running, use the terminal command `scheduler daemon start/stop`. This command only works if the scheduler is not disabled in the file `reportserver.properties`. If it is, you first have to delete this property in order to be able to enable/disable the scheduler while ReportServer is running. Refer to the ***Administration Guide*** for more information on this command.

Number of working threads

By default, ReportServer initializes a scheduling thread pool containing 5 threads. Consequently, a maximum of 5 scheduler jobs can run simultaneously. This setting can be modified by adjusting the following:

```
  <properties>
    <workingthreads>5</workingthreads>
```

```
</properties>
```

Thread pool name

Allows configuration of the scheduling thread pool's name. The default is `dwScheduler`.

```
<properties>
  <asyncpoolname>dwScheduler</asyncpoolname>
</properties>
```

Check interval

Allows configuration of the working thread's sleep time (in milliseconds) before checking for and executing the next set of scheduled jobs. The default is 10,000 milliseconds (10 seconds).

```
<properties>
  <checkinterval>10000</checkinterval>
</properties>
```

Shutdown timeout

This allows you to configure the timeout for a proper scheduler shutdown. It ensures that the scheduler blocks until one of the following occurs: all tasks have completed execution after a shutdown request, the specified timeout is reached, or the current thread is interrupted. The timeout is specified in milliseconds. The default value is 60,000 milliseconds (60 seconds).

This ensures that the scheduler attempts an orderly shutdown by waiting for the ongoing tasks to complete. However, if they do not finish within the timeout period, the scheduler forces an immediate shutdown to ensure the system does not hang indefinitely. This mechanism helps balance between graceful task completion and the need to shut down promptly.

```
<properties>
  <shutdownwait>600000</shutdownwait>
</properties>
```

Standard and random veto delay

When an execution veto occurs, such as with conditional scheduling, and when a RETRY is needed but no specific waiting period is defined, the system uses a combination of a default base waiting time and a random waiting time to determine the delay before the next retry.

The formula used to calculate the waiting time before the next RETRY is as follows:

```
int minutes = stdvetodelay + new Random().nextInt(rndvetodelay)
```

In this formula:

- `stdvetodelay` represents the standard veto delay, which is the base waiting time in minutes.
- `rndvetodelay` represents the random delay, also in minutes, which adds variability to the waiting time.

By default, `stdvetodelay` is set to 180 minutes, and `rndvetodelay` is set to 20 minutes. This means that the waiting time before the next retry will be at least 180 minutes, with an additional random component ranging from 0 to 20 minutes, making the total waiting time somewhere between 180 and 200 minutes.

Here is how you can configure these values in the properties file:

```
<properties>
  <stdvetodelay>180</stdvetodelay>
  <rndvetodelay>20</rndvetodelay>
</properties>
```

This configuration ensures that the system waits for a base period (`stdvetodelay`) and then adds a random delay (`rndvetodelay`) to determine the next retry time, helping to avoid predictable retry patterns and potential resource contention.

Further, refer to Section 4.13 Scheduler Settings for more scheduler settings.

4.5 Remote RS Server Settings

ReportServer 4.6.0 introduced new "Remote RS Server" objects as explained in the Administration Guide. Also, new `rpull` terminal commands were introduced in order to pull/copy remote entities, e.g. reports from a remote RS installation into the local RS installation, e.g. from PROD to TEST.

In order to be able to import remote reports, their datasources must be able to be mapped to local datasources. This mapping is defined via the `/etc/main/mappings.cf` configuration file explained next.

The default `/etc/main/mappings.cf` configuration file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <datasources>
      <priorities>
         <priority>mapping</priority>
         <priority>same-key</priority>
      </priorities>
      <key-mappings>
      <!--
         <key-mapping>
            <remote>REMOTE_KEY_1</remote>
            <local>LOCAL_KEY_1</local>
         </key-mapping>
         <key-mapping>
            <remote>REMOTE_KEY_2</remote>
            <local>LOCAL_KEY_2</local>
         </key-mapping>
      -->
      </key-mappings>
   </datasources>
</configuration>
```

The datasource mapping is performed via datasource keys by the key-mapping elements.

For example, the following configuration maps the remote datasource with key MY_REMOTE_DATASOURCE to the local datasource with key MY_LOCAL_DATASOURCE.

```
<key-mapping>
   <remote>MY_REMOTE_DATASOURCE</remote>
   <local>MY_LOCAL_DATASOURCE</local>
</key-mapping>
```

The priorities element defines the mapping priorities. There are currently two types of mappings:

- mapping: the explicit mapping defined by the key-mapping elements.
- same-key: implicit mapping of datasources with the same key.

For example, in the following configuration, ReportServer tries first to map the datasources via the explicit key-mapping elements. If no mapping is found, it tries to find a local datasource with the same key.

```
<priorities>
  <priority>mapping</priority>
  <priority>same-key</priority>
</priorities>
```

On the contrary, in the following configuration, ReportServer tries first to find a local datasource with the same key. If no implicit key mapping could be found, it tries to map the datasources via the explicit key-mapping elements.

```
<priorities>
    <priority>same-key</priority>
  <priority>mapping</priority>
</priorities>
```

4.6 Transport settings

As explained in the Administration Guide, in order to allow to use the transport mechanism, you have to configure ReportServer first. There are two relevant files to be configured:

- `main/mappings.cf`. This file is described in Section 4.5 and works analogously for transports for mapping remote datasources to local datasources.
- `main/transports.cf`. This file is described below.

This section explains the configuration options available in the `main/transport.cf` file, which is used for managing transports within ReportServer.

File Structure

The `main/transport.cf` file is an XML configuration file containing settings for importing and applying transports. Below is the breakdown of each element within the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <import>
      <remote>REMOTE_SERVER</remote>
      <target>/transports/import</target>
   </import>
   <apply>
      <!-- the username of the user that applies the transport. Must ↙
         ↳ be a super user. -->
      <user>root</user>
   </apply>
</configuration>
```

Elements and Their Descriptions

`<configuration>`

The root element that encapsulates all configuration settings for the transport process.

`<import>`

This section contains settings related to the import of transports from a remote server.

`<remote>`	Defines the identifier for the remote server from which transports are imported. This should match the key configured for the remote server in ReportServer.
`<target>`	Specifies the directory path in the ReportServer's file system where the imported transports are stored in the local ReportServer after importing them from the remote server. If the path does not exist, it is created automatically by ReportServer during the import process.

`<apply>`

`<user>`	Specifies the username of the user who applies the transport. The user must have superuser privileges in ReportServer to perform this action.

Summary

- The `transport.cf` file is used to configure the import of transports from a remote server and the application of those transports within ReportServer.
- Ensure that the `<remote>` value correctly identifies the remote server, and that the `<target>` directory is properly set up with the correct permissions.
- The `<user>` applying the transport must be a superuser to ensure successful integration into the target system.

4.7 Report execution error log settings

You may configure the exact detailed information your ReportServer should log in case of report execution failure. This detailed information can be set in the `main/error-log.cf` configuration file.

The results are printed in a JSON string for easy result analysis. Note that the order of the properties is preserved in the result's JSON string.

Per default, ReportServer logs the following information:

`error`	a description of the root error causing the failure
`report_id`	the ID of the report
`report_name`	the name of the report
`base_report_id`	the ID of the base report (in case of variants)
`base_report_name`	the name of the base report (in case of variants)
`executing_user_id`	the ID of the user trying to execute the report
`report_output_format`	the output format of the report (e.g. `HTML`, `PDF`, etc)
`report_uuid`	the UUID (Universally Unique Identifier) of the report

Available are the properties discussed below.

Error properties

The following properties allow you to print information of the root error.

`error`	a description of the root error causing the failure

Report properties

The following properties allow you to print information of the report which is failing.

report_id	the ID of the report
report_name	the name of the report
report_key	the key of the report
report_type	the type of the report, e.g. TableReportVariant, JasperReport, etc
base_report_id	the ID of the base report (in case of variants)
base_report_name	the name of the base report (in case of variants)
base_report_key	the key of the base report (in case of variants)
base_report_type	the type of the base report (in case of variants), e.g. TableReportVariant, JasperReport, etc
report_output_format	the output format of the report (e.g. HTML, PDF, etc)
report_uuid	the UUID (Universally Unique Identifier) of the report

User properties

The following properties allow you to print information of the user trying to execute the report which is failing.

executing_user_id	the ID of the user trying to execute the report
executing_user_username	the username of the user trying to execute the report
executing_user_firstname	the first name of the user trying to execute the report
executing_user_lastname	the last name of the user trying to execute the report

Datasource properties

The following properties allow you to print information of the failing report's datasource.

datasource_id	the ID of the report's datasource
datasource_name	the name of the report's datasource
datasource_path	the path of the report's datasource in the fileserver file system
datasource_type	the type of the report's datasource, e.g. DatabaseDatasource, ScriptDatasource, etc

In addition to the properties above, you can use the following for DatabaseDatasources (i.e. for relational databases).

`datasource_database_query`	the report's datasource query. Note that the query is printed exactly as entered into the datasource's "query" field, i.e. no parameter replacement is shown in the output.
`datasource_database_information`	general information of the report's datasource. This information includes the following: • `databaseProductName`: the database product, e.g. "MySQL", "H2", etc. • `databaseProductVersion`: the version of the database, e.g. "8.0.31" • `driverName`: the JDBC driver, e.g. "MySQL Connector/J" • `driverVersion`: the JDBC driver's version: "mysql-connector-j-8.0.31" • `JDBCMajorVersion`: the JDBC major version of the driver • `JDBCMinorVersion`: the JDBC minor version of the driver • `URL`: the datasource's complete JDBC URL • `userName`: the datasource's JDBC username
`datasource_database_jdbc_properties`	the JDBC properties of the report's datasource.

General properties

The following properties allow you to print general information of your ReportServer's installation. Basically, this is the same information you may find in your "General Info" system console.

java_version	your exact Java version
java_home	environment variable pointing to the your Java installation directory
java_vm_arguments	the Java Virtual Machine arguments
application_server	your application server, e.g. "Apache Tomcat/9.0.68"
catalina_home	the root of your application server installation, for example "/home/tomcat/apache-tomcat-9.0.10" or "C:/Program Files/apache-tomcat-9.0.10"
catalina_home	the root of a runtime configuration of a specific application server instance
log_files_directory	path to the directory where the log files are located
rest_url	the REST access point
request_url	the complete request URL
request_scheme	the name of the scheme used to make the request, for example, "http", "https"
request_server_name	the host name of the server to which the request was sent
request_server_port	the port number to which the request was sent
request_context_path	the portion of the request URI that indicates the context of the request. The context path always comes first in a request URI. The path starts with a "/" character but does not end with a "/" character. For servlets in the default (root) context, this method returns "".
request_protocol	the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion, for example, "HTTP/1.1"
config_directory	path to your configured external configuration directory
os_version	your operation system, e.g. "Windows Server 2022"
reportserver_version	your ReportServer's complete version
locale	your current ReportServer's locale
jvm_locale	the locale of your Java Virtual Machine
jvm_user_language	the language of your Java Virtual Machine user, e.g. "de"
jvm_user_country	the country of your Java Virtual Machine user, e.g. "DE"
jvm_user_timezone	the timezone of your Java Virtual Machine user, e.g. "Europe/Berlin"
jvm_file_encoding	the file encoding of your Java Virtual Machine user, e.g. "UTF-8"
known_hosts_file	path to your configured known_hosts file
supported_ssl_protocols	the supported SSL protocols for the current SSL context
default_ssl_protocols	the default SSL protocols for the current SSL context
enabled_ssl_protocols	the enabled SSL protocols for the current SSL context
groovy_version	your exact Groovy version
active_pams	a list of your installed active PAMs (Pluggable Authentication Modules)

Memory properties

The following properties allow you to print information of the current memory settings during the report's execution failure. This is basically the same information your `meminfo` terminal command prints.

`memory_used`	your current memory usage
`memory_free`	the amount of free memory in the Java Virtual Machine
`memory_total`	the total amount of memory in the Java Virtual Machine. The value returned by this method may vary over time, depending on the host environment.
`memory_max`	the maximum amount of memory that the Java virtual machine will attempt to use.

4.8 Export settings

Regarding the export to PDF and Microsoft Excel, there are several options that you can set. In `/fileserver/etc/exportfilecmd/excelexport.cf` you can specify in which format Excel documents are to be exported. You can choose between the old XLS and the current XLSX format. If you have selected the XLSX format, ReportServer allows to stream the data to the client. This means, that the chunks of resulting Excel file are sent to the user while it is still being created. Streaming result files can significantly reduce processing time.

To specify the Excel format and whether or not to use streaming, adjust the following parameters:

```
<format>xlsx</format>
<stream>true</stream>
```

Further, you can specify the title of the data and configuration sheets in the resulting Excel file by the following parameters:

```
<datasheet>Dynamic list</datasheet>
<configsheet>Configuration</configsheet>
```

When exporting to Excel, you can organize the data into multiple tabs by setting the number of rows per tab. The default value is 1,048,574, which is selected because Excel allows a maximum of 1,048,576 rows. The difference accounts for one row used by the headings and one for the final newline. To determine the number of rows per sheet, modify the following parameter:

```
<maxrecordspertab>100</maxrecordspertab>
```

Refer to the "output_parameters", "output_filters", and "output_complete_configuration" report parameters in the Admin Guide for more information on this.

You can specify document properties (title, creator and author) of PDF files in the configuration file `/fileserver/etc/exportfilecmd/metadata.cf`. The texts specified here will be included in newly generated PDF files.

Modify the following parameters:

```
<title>title</title>
<creator>ReportServer</creator>
<author>ReportServer</author>
```

As with email notifications you can use substitutions to dynamically populate the fields. The following substitutions are available:

Available Substitutions

Expression	Description
`${user.getUsername()}`	username
`${user.getFirstname()}`	user's first name
`${user.getLastname()}`	user's last name
`${user.getEmail()}`	user's email address
`${user.getTitle()}`	user's title
`${user.getId()}`	user's id

You can further specify the default character set used by ReportServer. In the configuration file `/fileserver/etc/main/main.cf` you will find the option charset. By default, the charset UTF-8 is used.

Dynamic list export thresholds

When exporting a dynamic list, ReportServer allows to configure two different thresholds. If the export contains more records than the first threshold, the user gets a warning when trying to export the report. If the export contains more records than the second threshold, the export is being prohibited.

Both thresholds can be configured in `/main/main.cf`:

```
<export>
   <warnthreshold>10000</warnthreshold>
   <maximumrecords>100000</maximumrecords>
</export>
```

4.9 UI Customization

In this section we cover the possibilities of customizing the user interface. ReportServer provides the following customization options:

- Specifying the default language
- Customizing error messages
- Customizing the theme
- Customizing the PDF preview
- Customizing the preview of the dynamic list
- Customizing the report documentation
- Adding tabs based on context

Further customization options are available with the use of ReportServer scripts. More information on ReportServer scripts can be found in the ***Administration Guide***.

Specifying the available languages

Any visible text in ReportServer can, in principle, be displayed in any language. The languages available on log-in can be defined in the configuration file `/fileserver/etc/main/localization.cf`.

```
<default>de</default>
```

The "default" property specifies which language to use as default language.

```
<locales>en,fr,de</locales>
```

The "locales" property specifies a comma-separated list of available languages. If the property is not specified, all supported languages are available for selection.

Remark. The user's selection is stored in a cookie. Thus, the change of the default locale will not override any locale settings done by a user previously.

Remark. A large part of the translations have been generated in a semi-automatic way and are thus far from perfect. If you are a native speaker in one of the languages and would like to contribute please contact us at info@infofabrik.de.

Customization of error messages

Errors can occur due to various reasons.

Typical errors are:

- an error occurs on the database during report execution, because:
  - a table does not exist,
  - a column does not exist,
  - there is a syntax error in the underlying SQL,
  - the JDBC driver was not installed
  - etc.
- the connection pool does not have any free connections

An exception is thrown whenever an error occurs in ReportServer. The exception is composed of: a title, error message, and the stack trace.

In `/fileserver/etc/main/templates.cf` you can customize the error message that is displayed on errors that occur during the export of reports.

When customizing the error message you should give clear instructions as to what the affected employee should do in this case. Usually you would specify the contact address of an administrator or help desk. When customizing, the following substitutions are available: `${headline}`, `${msg}` and `${stacktrace}`.

Customization of theming

In ReportServer Enterprise Edition it is possible to customize the theme via the `/fileserver/ui/theme.cf` config file. Further information on this can be found in the administration guide.

Preview for PDF reports

In the configuration file `/fileserver/etc/ui/previews.cf` you can specify how to render PDF previews. The options are `${native}` (to use the native browser capabilities), `${jsviewer}` (to use a javascript library) or `${image}` to not render a PDF at all, but to only render the first page as an image. Also note that users can overwrite the settings within their profiles.

Customizing the preview of the dynamic list

The configuration file `/fileserver/etc/ui/previews.cf` allows you to configure the preview of the dynamic list. The complete default `dynamicList` section is shown below:

```
<dynamicList>
   <defaultColumnWidth>200</defaultColumnWidth>
   <maxColumnWidth>800</maxColumnWidth>
   <pageSize>
      <configs>
         <config minCols = "0" maxCols = "99">50</config>
         <config minCols = "100" maxCols = "249">25</config>
         <config minCols = "250" maxCols = "499">10</config>
         <config minCols = "500" maxCols = "MAX">5</config>
      </configs>
   </pageSize>
</dynamicList>
```

The `defaultColumnWidth` setting allows you to configure the global default width of the dynamic list columns. Note that the width of a column may be configured individually in the variant configuration screen. If no width is configured there, the default global width is used. The `maxColumnWidth` allows you to configure the maximum column width.

The number of rows of the dynamic list preview can be configured in the `pageSize` section. The number of rows is dependent of the number of columns selected. In the example above, the preview will have:

- 50 rows if the number of colums is greater equal 0 or less equal 99
- 25 rows if the number of colums is greater equal 100 or less equal 249
- 10 rows if the number of colums is greater equal 250 or less equal 499

- 5 rows if the number of colums is greater equal 500

Note that you can use the `MAX` keyword for denoting the maximum number of columns.

In case you don't want the number of rows to be dependent on the number of columns, you can configure it as follows:

```
<pageSize>
   <configs>
      <config minCols = "0" maxCols = "MAX">50</<config>
   </configs>
</pageSize>
```

Adding contextual tabs

Using the configuration file `/fileserver/etc/ui/urlview.cf` you can define context aware tabs to be displayed in the TeamSpace or in the administration module (e.g., report management, user management, etc.). This allows you to, for example, display the documentation report directly whenever a user selects a report in the TeamSpace.

The configuration of extra tabs is split into three parts:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <adminviews>

  </adminviews>
  <objectinfo>

  </objectinfo>
  <module>

  </module>

</configuration>
```

Tabs to be displayed in the administration module go into the **adminviews** tags. Tabs for the TeamSpace are put within the **objectinfo** tags. Tabs for ReportServer modules are put within the **module** tags. The default configuration does not add any additional tabs for the admin interface or to the modules, but adds several tabs to the TeamSpace:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <adminviews>
   </adminviews>
   <objectinfo>
     <view>
       <types>net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
           ↳ TsDiskReportReferenceDto</types>
       <name>${msgs['net.datenwerke.rs.core.service.urlview.locale.UrlViewMessages'][']↙
           ↳ info']}</name>
       <url>rs:reportdoc://${reportId}/${id}</url>
     </view>
     <view>
```

```
      <types>net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
          ↳ TsDiskReportReferenceDto</types>
      <name>${msgs['net.datenwerke.rs.core.service.urlview.locale.UrlViewMessages']['↙
          ↳ history']}</name>
      <url>rs:revisions://${reportId}</url>
     </view>
     <view>
      <types>net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
          ↳ TsDiskReportReferenceDto</types>
      <name>${msgs['net.datenwerke.rs.core.service.urlview.locale.UrlViewMessages']['↙
          ↳ preview']}</name>
      <url>rs:reportpreview://${reportId}</url>
     </view>
   </objectinfo>
   <module>
   </module>
</configuration>
```

Each <view> tag adds a new tab. The <types> tag allows to define for which types of objects the tab is displayed and <name> provides a name (in the above example, the name is localized, but you could also simply write <name>SomeName</name>. Finally, the <url> tag takes a URL that is to be displayed. In the above example we have three custom ReportServer URLs that access custom functionality, the first accesses a documentation report, the second a revisions report and the last one a preview of the report. Via the replacement ${reportId} the id of the object is added to the URL.

In the following we go through the process of adding new tabs step by step.

Adding a new Tab

To add a new tab, you define a <view> tag. For adding a tab to the TeamSpace whenever a report is selected add the following <view> tag within the <objectinfo> section.

```
  <view>
    <types>
      net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
          ↳ TsDiskReportReferenceDto
    </types>
    <name>Some Additional Information</name>
    <url>
      reportserver/reportexport?key=someKey&amp;format=html&amp;↙
          ↳ p_reportId=${reportId}
    </url>
  </view>
```

The above will execute the report with key "***someKey***" and pass the given report id as parameter.

The following types are available in TeamSpaces.

All Objects in a TeamSpace:

```
net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.AbstractTsDiskNodeDto
```

All folders in a TeamSpace:

`net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.TsDiskFolderDto`

All reports and variants in a TeamSpace:

`net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.TsDiskReportReferenceDto`

All ***exported reports*** which were, for example, created by the scheduler:

`net.datenwerke.rs.scheduleasfile.client.scheduleasfile.dto.ExecutedReportFileReferenceDto`

The name field defines the tab's name. The url is the address that is displayed. This also allows you to access external addresses that are then displayed within the tab.

If you need to restrict the tab to some users, groups or OUs, you can achieve this with help of the `restrictTo` tag. For example, if you want to restrict the tab shown above to users with "demoadmin" and "demoadmin2" `username`s, you can enter these usernames separated by commas as shown below:

```
<view>
  <types>
    net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.
        TsDiskReportReferenceDto
  </types>
  <name>Some Additional Information</name>
  <url>
    reportserver/reportexport?key=someKey&amp;format=html&amp;
        p_reportId=${reportId}
  </url>
  <restrictTo>
    <users>demoadmin1,demoadmin2</users>
  </restrictTo>
</view>
```

If you need to restrict the tab shown above to groups with `id`s "1" and "2", you can enter them separated by commas as shown below:

```
<view>
  <types>
    net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.
        TsDiskReportReferenceDto
  </types>
  <name>Some Additional Information</name>
  <url>
    reportserver/reportexport?key=someKey&amp;format=html&amp;
        p_reportId=${reportId}
  </url>
  <restrictTo>
    <groups>1,2</groups>
  </restrictTo>
</view>
```

The sample above applies also to organizational units. For restricting the tab to organizational units with ids "3" and "4", you can enter them separated by commas as shown below:

```
<view>
  <types>
    net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
        ↳ TsDiskReportReferenceDto
  </types>
  <name>Some Additional Information</name>
  <url>
    reportserver/reportexport?key=someKey&amp;format=html&amp;↙
        ↳ p_reportId=${reportId}
  </url>
  <restrictTo>
    <ous>3,4</ous>
  </restrictTo>
</view>
```

Note that `restrictTo` needs comma-separated `username`s for users and comma-separated `id`s for groups and organizational units.

Combining restrictions is also possible. The following configuration restricts the tab to "demoadmin" and "demoadmin2" users or "1" and "2" groups. This means, the uses "demoadmin", "demoadmin2", or any user in the "1" and "2" groups can access the tab.

```
<view>
  <types>
    net.datenwerke.rs.tsreportarea.client.tsreportarea.dto.↙
        ↳ TsDiskReportReferenceDto
  </types>
  <name>Some Additional Information</name>
  <url>
    reportserver/reportexport?key=someKey&amp;format=html&amp;↙
        ↳ p_reportId=${reportId}
  </url>
  <restrictTo>
    <users>demoadmin1,demoadmin2</users>
    <groups>1,2</groups>
  </restrictTo>
</view>
```

For the report documentation you need to use the special ReportServer URL:

```
rs:reportdoc://${reportId}/${id}
```

As you can see there are two placeholders in the above url: `reportId` and `id`. The following replacements are available

id	the object's id
type	the object's type
username	the current user's username

Note that TeamSpaces do not only contain report references. Thus, the replacement `${id}` will contain the id of the reference rather than the id of the referenced report. For this, there is the special replacement called `${reportId}` which is only available for report references.

Similarly, to the report documentation in the TeamSpace you can display additional information on any selected object in the ***administration module***. In the administration module you can add tabs to objects in the report management, user management, dadget management, datasource management and fileserver modules. These are configured within the `<adminviews>` tag.

The tables in the following subsections describe which types can be used. Note that the type must be used together with the corresponding prefix.

Objects in Report Management

Prefix `net.datenwerke.rs.core.client.reportmanager.dto.reports`.

Type	Description
AbstractReportManagerNodeDto	All objects in the report management tree
ReportDto	reports
ReportFolderDto	folders

Objects in User Management

Prefix `net.datenwerke.security.client.usermanager.dto`.

Type	Description
AbstractUserManagerNodeDto	All objects in the user management tree
UserDto	users
GroupDto	groups
OrganisationalUnitDto	organisational units (folders)

Objects in the file server

Prefix `net.datenwerke.rs.fileserver.client.fileserver.dto`.

Type	Description
AbstractFileServerNodeDto	All objects in the file server
FileServerFolderDto	folders
FileServerFileDto	files

Objects in Datasource Management

Prefix `net.datenwerke.rs.core.client.datasourcemanager.dto`.

Type	Description
AbstractDatasourceManagerNodeDto	all objects in datasource management
DatasourceFolderDto	folders
DatasourceDefinitionDto	datasources

Objects in Dadget Management

Prefix net.datenwerke.rs.dashboard.client.dashboard.dto.

Type	Description
AbstractDashboardManagerNodeDto	All objects in the dashboard tree
DashboardNodeDto	dashboards
DashboardFolderDto	folders

The following example would display a tab **User Information** which displays the website at Url http://www.mycompany.com/employee.

```
<adminviews>
  <view>
    <types>net.datenwerke.security.client.usermanager.dto.UserDto</↙
       ↪ types>
    <name>User Information</name>
    <url>http://www.mycompany.com/employee=${id}</url>
  </view>
</adminviews>
```

Adding modules

As explained in Section 4.9, you can define context-aware tabs to be displayed in the TeamSpace or in the administration module.

In addition, you can introduce custom modules to ReportServer, which will be visible in the module tab section at the top of ReportServer. Essentially, the new module will appear after the "Administration" module.

Here's an example configuration for adding new modules:

```
<module>
   <view>
       <name>Section A</name>
       <url>http://SERVER:PORT/reportserverbasedir/reportserver/↙
          ↪ scriptAccess?id=15</url>
    </view>
    <view>
       <name>Section B</name>
       <url>http://SERVER:PORT/reportserverbasedir/reportserver/↙
          ↪ scriptAccess?id=16</url>
   </view>
</module>
```

In this example, two new modules, Section A" and Section B", are defined. These modules link to scripts in ReportServer (with id 15 and 16, respectively). While you can specify any URL within the configuration, linking to ReportServer scripts generally provide a higher level of customization.

The restrictTo discussed in Section 4.9 also applies here. The following restricts the "Section A" tab to groups with "1" and "2" ids:

```
<module>
```

```
        <view>
            <name>Section A</name>
            <url>http://SERVER:PORT/reportserverbasedir/reportserver/↙
                ↳ scriptAccess?id=15</url>
            <restrictTo>
               <groups>1,2</groups>
        </restrictTo>
         </view>
         <view>
            <name>Section B</name>
            <url>http://SERVER:PORT/reportserverbasedir/reportserver/↙
                ↳ scriptAccess?id=16</url>
        </view>
    </module>
```

Customization of the Login Page

Following is a quick guide for those who want to completely exchange the ReportServer login page by a custom looking page. In the following we assume that the FileServer contains a folder /resources/public and that the public folder is marked as "web accessible". In case not, you can either mark it with the "web accessible" checkbox in the UI, or use the following commands:

```
cd /fileserver/resources
dirmod webaccess public true
```

As a first step we are going to create a very simple login page, something along the following lines:

```
<html>
   <head>
      <title>Custom Login</title>
   </head>
   <body>
      My custom login page
      <form method="post" action="">
         <label for="user">username:</label>
         <input type="text" name="user" /> <br/>
         <label for="pw">password:</label>
         <input type="password" name="pw" /></br>
         <input type="submit"/>
      </form>
   </body>
</html>
```

The page consists of a single form with a username and password field. We store the above as login.html in the folder /resources/public. For this, for example open the terminal (CTRL+ALT+T) and go for

```
cd /fileserver/resources/public
createTextFile login.html
```

As the public folder is accessible for anybody (even if the user is not logged in) we can access it via the fileServerAccess servlet. That is, if your ReportServer is accessible via the url

reporting.mycompany.com/ then you should see the login page if you go to http://reporting.mycompany.com/reportserver/fileServerAccess?path=/resources/public/login.html.

Now what was missing in the above login page was the intended target of the form. For this we will create a publicly accessible script that handles the form data. As scripts always go into the bin folder we create a folder public beneath the bin folder and add a script customauth.groovy.

```
cd /fileserver/bin
mkdir public
cd public
createTextFile customauth.groovy
dirmod webaccess public true
```

To see that everything worked, we use the following simple script which simply outputs the parameter user:

```
def user = httpRequest.getParameter('user')
return user
```

Note that you have access to the request and response via the httpRequest and httpResponse variables. What is left is to change the action attribute of the login.html page to point to the script, that is, we need to change it to http://reporting.mycompany.com/reportserver/scriptAccess?path=/bin/public/customauth.groovy.

```
<html>
   <head>
      <title>Custom Login</title>
   </head>
   <body>
      My custom login page
      <form method="post" action="http://reporting.mycompany.com/↙
         ↳ ReportServer/reportserver/scriptAccess?path=/bin/public/↙
         ↳ customauth.groovy">
         <label for="user">username:</label>
         <input type="text" name="user" /> <br/>
         <label for="pw">password:</label>
         <input type="password" name="pw" /></br>
         <input type="submit"/>
      </form>
   </body>
</html>
```

The Authentication

What we have so far is that we have a custom login page and a script which is the target. What we need is that the script can actually perform the login operation if the provided credentials match. For this we will use the AuthenticatorService located in

```
net.datenwerke.security.service.authenticator.AuthenticatorService
```

In the following we check for a username root and a password 123. If found we perform a login for user with id 6 (which in my case is the root user).

```
import net.datenwerke.security.service.authenticator.↙
    ↳ AuthenticatorService

def service = GLOBALS.getInstance(AuthenticatorService).get()

def user = httpRequest.getParameter('user')
def pw = httpRequest.getParameter('pw')

if( 'root' == user && '123' == pw ){
  service.setAuthenticated(3) // the id of the root user
  httpResponse.sendRedirect('http://reporting.mycompany.com/↙
      ↳ ReportServer/ReportServer.html');
  return null
} else {
  return 'Could not authenticate'
}
```

If all went well you can now logoff and log in via the custom page http://reporting.mycompany.com/reportserver/fileServerAccess?path=/resources/public/login.html.

Authentication against the ReportServer User Database

In the above example we had a custom script to handle the authentication of a single user. This of course does not scale well and for any real scenario one would like to authenticate against a user database. In the following we show how to authenticate against ReportServer's own user database. For this we can again use the AuthenticatorService which offers a method authenticate that triggers the built-in authentication mechanisms.

ReportServer's built-in authentication is structured in so called pluggable authentication modules (short PAM) which perform the actual authentication. The active PAMs are configured in the reportserver.properties configuration file and usually only a single PAM is active:

```
rs.authenticator.pams = net.datenwerke.rs.authenticator.service.pam.↙
    ↳ UserPasswordPAMAuthoritative
```

The above config loads the UserPasswordPAM module in authoritative mode (more information on the PAMs can be found in the configuration guide). This PAM expects a username and password and then sets of to authenticate against the ReportServer user database. The authoritative flag means that if the UserPasswordPAM cannot authenticate a user that it will then trigger an abort. This can become necessary when you would like to combine multiple different PAMs.

As explained, the AuthenticatorService's authenticate method triggers the internal authentication process. That is, it expects an array of so called AuthTokens (which can be basically anything) and then hands these to the registered PAMs. The PAMs are then asked in turn whether or not they can authenticate a user. For this they use the AuthToken array. The UserPasswordPAM thus expects a username and password as an AuthToken. This is encapsulated in the UserPasswordAuthToken which is located in

```
net.datenwerke.rs.authenticator.client.login.dto.UserPasswordAuthToken
```

The authenticate method returns an AuthenticationResult which can be asked whether the authentication succeeded (`isAllowed()`). In the following script we combine our earlier example with an authentication against ReportServer's user database.

```
import net.datenwerke.security.service.authenticator.↙
    ↳ AuthenticatorService
import net.datenwerke.security.client.login.AuthToken
import net.datenwerke.rs.authenticator.client.login.dto.↙
    ↳ UserPasswordAuthToken

def service = GLOBALS.getInstance(AuthenticatorService)

def user = httpRequest.getParameter('user')
def pw = httpRequest.getParameter('pw')

/* construct authentication tokens */
def token = new UserPasswordAuthToken()
token.username = user
token.password = pw

def result = service.authenticate([token] as AuthToken[])

if(result.isAllowed()){
  httpResponse.sendRedirect('http://reporting.mycompany.com/↙
      ↳ ReportServer/ReportServer.html')
  return null
}

return 'Could not authenticate'
```

Now your custom login page should be fully functional. The logout part, explained next, should be customized as well.

Custom Logout

Currently when a user logs out, the user will be taken back to the original ReportServer login page. In order to change that we need to change the config file located in `etc/security/misc.cf` within the ReportServer filesystem. If we add

```
<logout>
   <url>http://reporting.mycompany.com/reportserver/fileServerAccess?↙
       ↳ path=/resources/public/login.html</url>
</logout>
```

to the config, then on logout ReportServer will redirect the user to http://reporting.mycompany.com/reportserver/fileServerAccess?path=/resources/public/login.html.

The complete example can found here: https://github.com/infofabrik/reportserver-samples/tree/main/src/net/datenwerke/rs/samples/admin/login/simple.

Advanced example

Based on the previous examples, you can used advanced techniques, together with e.g. JQuery: https://jquery.com/ to further customize your login page. An example of this can be found here: https://github.com/infofabrik/reportserver-samples/tree/main/src/net/datenwerke/rs/samples/admin/login/jquery.

4.10 Extensions

ReportServer has a modular design which is exposed in ReportServer Enterprise Edition to allow for customization. The extension points are called ***hooks***. Extensions are written in groovy (https://groovy-lang.org/) and can ***hook into*** various places in ReportServer. In order to use scripts, you must configure certain properties in the configuration file `/fileserver/etc/scripting/scripting.cf`.

This file controls whether scripts are enabled to begin with. Furthermore, you have to specify a path (in the internal filesystem) beneath which scripts can be placed (this helps to allow users to create/edit files in the file system without giving them the rights to write scripts). Finally, you can name a script which is executed on ReportServer startup and one which is executed whenever a user logs in.

```
<scripting>
  <enable>true</enable>
  <restrict>
    <location>bin</location>
  </restrict>
  <startup>
    <login>fileserver/bin/onlogin.groovy</login>
    <rs>fileserver/bin/onstartup.groovy</rs>
  </startup>
</scripting>
```

In the above example we allow scripts only in the bin folder (and subfolders). We defined the script `fileserver/bin/onlogin.groovy` as the script that is executed whenever a user logs in (note that the script is executed with the current user, that is, the user that logged in and thus the user must have the rights to execute this script). The second, on startup script is executed without any user.

The ***onstartup*** and ***onlogin*** scripts shipped with the ReportServer demo data allow you to easily execute your own scripts. The ***onstartup*** script executes all scripts in the folder `/fileserver/bin/onstartup.d`. Likewise, the `onlogin.groovy` script executes all scripts within `/fileserver/bin/onlogin.d`.

Further information on ReportServer scripts can be found in the ***administration guide*** and the specialized ***scripting guide***.

4.11 Executing Reports using URLs

You can configure ReportServer to allow other applications to call reports or to include reports into external websites. For this ReportServer allows to call and configure reports directly using a

specific URL. You can find more information about how reports can be accessed using a URL in the ***administrators guide***.

The old httpauthexec functionality was replaced in ReportServer 3.0. See the Administration Guide for further information on how to make reports available via the URL without login.

4.12 Misc Settings

ReportServer needs a directory to store temporary files. You can define the directory to use in `/fileserver/etc/main/main.cf`. Furthermore, you can specify a maximum lifetime (in seconds) of temporary files in the directory.

```
<tempdir>tempdir</tempdir>
<tempfile>
  <lifetime>3600</lifetime>
</tempfile>
```

Maintenance Tasks

ReportServer performs certain maintenance tasks from time to time. You can define the interval (in ms) with which ReportServer will execute these.

```
<maintenance>
  <tasks>
    <interval>600000</interval>
  </tasks>
</maintenance>
```

ReportServer allows to define a timeout (in ms) for search queries:

```
<search>
  <timeout>5000</timeout>
</search>
```

4.13 Scheduler Settings

While scheduling new jobs, you can select the option "Compress report" if you want the report to be compressed in the email sent. Normally, this option is not selected by default. Now you can set this setting to be selected by default (in `/fileserver/etc/main/main.cf`):

```
<scheduler>
  <email>
    <defaultcompression>true</defaultcompression>
  </email>
</scheduler>
```

If you set this to `true`, "Report compression" will be automatically selected when scheduling new jobs. Defaults to `false`.

Refer to Section 4.4 Scheduler settings for more scheduler settings.

4.14 Localization Settings

ReportServer contains localization settings in the `/fileserver/etc/main/localization.cf` file.

You can configure the languages available to your ReportServer users with the following:

```
<locales>en,fr,de</locales>
```

In the example above, english, french and german languages are enabled. The default language can be configured with the following setting:

```
<default>fr</default>
```

In the example above, only french will be auto-selected in the ReportServer language list. Note that for users that previously used ReportServer, the language they used is saved in a cookie for next time, so the default language setting will not have any effect on these users. You have to delete your browser's cookies for this.

You can find more information on the available language codes here: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Locale.html.

The country/region (note the uppercase) in which you are using your ReportServer can be configured as follows:

```
<region>US</region>
```

This may be important when using currencies in your reports. For example, if your Jasper reports are executed using the "DE" (Germany) region code, their currencies will be printed in euro. If they are executed using the "US" region code, their currencies will be printed in dollars.

You can find more information on the available country/region codes here: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Locale.html.

Formats can be configured in the following:

```
<format>
  <!--
    <shortDatePattern></shortDatePattern>
    <longDatePattern></longDatePattern>
    <shortTimePattern></shortTimePattern>
    <longTimePattern></longTimePattern>
    <shortDateTimePattern></shortDateTimePattern>
    <longDateTimePattern></longDateTimePattern>
    <numberPattern></numberPattern>
    <currencyPattern></currencyPattern>
    <integerPattern></integerPattern>
    <percentPattern></percentPattern>
  -->
</format>
```

For example, the shortDatePattern and numberPattern may be configured as follows:

```
<format>
  <shortDatePattern>y-MM-dd</shortDatePattern>
```

```
  <numberPattern># ##0,00</numberPattern>
</format>
```

Details on the formats available may be found here:

Number Formats http://www.gwtproject.org/javadoc/latest/com/google/gwt/i18n/client/NumberFormat.html

Date Formats http://www.gwtproject.org/javadoc/latest/com/google/gwt/i18n/client/DateTimeFormat.html

The currency locales may be configured in the following section:

```
<currencies>
        <currency language="de" region="DE">currencyEuro</currency>
        <currency language="en" region="US">currencyDollar</currency>
        <currency language="en" region="GB">currencyPound</currency>
        <currency language="ar" region="AE">AED</currency>
        <currency language="ps" region="AF">AFN</currency>
        ...
</currencies>
```

In the example above, the Euro currency is localized to the de_DE locale. If you need to change this, e.g. to fr_FR, you may change this to:

```
<currencies>
        <currency language="fr" region="FR">currencyEuro</currency>
        <currency language="en" region="US">currencyDollar</currency>
        <currency language="en" region="GB">currencyPound</currency>
        <currency language="ar" region="AE">AED</currency>
        <currency language="ps" region="AF">AFN</currency>
        ...
</currencies>
```

As currency is locale-specific, the format may change depending on the locale configured here.

For example, 123456.79 dollars will be printed as follows in the default locale:

```
US$123,456.79
```

In en_US locale, the same will be printed as:

```
$123,456.79
```

Note that you have to restart ReportServer if you change your currency locale configuration.

More details on currency locales may be found here: http://www.gwtproject.org/javadoc/latest/com/google/gwt/i18n/client/NumberFormat.html.

4.15 Security related properties

In the following section we describe ***certain security related*** configuration options.

In `/fileserver/etc/security/misc.cf` you can define a blacklist for ReportServer expressions. If not running ReportServer with a SecurityManager, you should ensure that such expressions cannot use java reflection. At a minimum level you should deny the phrase `getClass`. Further information can be found in the ***administration*** and ***user guides***.

```
<juel>
  <expression>
    <blacklist>getClass</blacklist>
  </expression>
</juel>
```

In the example the expression `getClass` is prohibited. Multiple expressions are comma separated.

URL Whitelist

Certain operations, e.g. redirect, are not allowed for external URLs. If you need to allow a URL you can add it to the whitelist in the `/fileserver/etc/security/whitelist.cf` configuration file.

```
<urls>
  <url>http://www.host.com</url>
</urls>
```

Configuring error message level of detail

Further, in this configuration file, you can select the level of detail of error messages shown to the user. Currently, the following is supported:

hideViolatedSecurityExceptionDetails When a security rule is violated, e.g. when a user tries to execute a report the user is not allowed to, specific details of the security violation are being printed to the user by default. This includes the specific security target, the rights/permissions being violated, the objects and the method where this happens, if available. This helps administrators to exactly understand which rights/permissions are needed for a specific object to be accessed by a user. If not desired, you can set this to false in order to hide these specific details from the user, which may be preferable in some cases. Defaults to false.

Below you can see an example configuration.

```
<errorMessages>
   <hideViolatedSecurityExceptionDetails>false</⤶
      ↳ hideViolatedSecurityExceptionDetails>
</errorMessages>
```

Disabling forgot password

If you wish to disable the “forgot password” option in the login window, you can set the ‘disableLost-Password” option to true.

Below you can see an example configuration for this.

```
<disableLostPassword>true</disableLostPassword>
```

Configuring cryptography

The file `/fileserver/etc/security/crypto.cf` defines various cryptography related options. The `<cryptocredentials>` section defines how cryptographic credentials, such as private keys and certificates are retrieved for various ReportServer modules.

To do so, a provider is specified for each module.

A provider is defined by specifying the name of the handler-class and some additional attributes.

```
<provider type="signature">
  <class>
    net.datenwerke.rs.incubator.service.crypto.↙
        ↳ FileServerKeyStoreKryptoCredentialProvider
  </class>
  <alias>rs</alias>
  <secret>secret</secret>
  <type>jks</type>
  <location>/fileserver/keystore.jks</location>
</provider>
```

This configures the default handler, which tries to load key-material from a file within the fileserver. Providers can be specified for these types:

signature: a keystore that holds the private key, ReportServer uses when sending signed emails

user: a keystore that holds public keys and certificates of ReportServer users. This is used, when sending encrypted emails. An alternate method to provide key material is by using a custom script, that retrieves the keys e.g. from a corporate directory.

Specifying a Password Policy

ReportServer allows to configure the use of password policies to ensure that users choose secure passwords. The corresponding configuration goes into the configuration file `/fileserver/etc/security/passwordpolicy.cf`.

For example, you can define how long passwords should be and define character classes from which the password must be built. Furthermore, you can define how often passwords need to be changed and when a previously chosen password may be chosen again.

```
<pswd>
  <maxage>32</maxage>
  <minage>1</minage>
  <minlength>8</minlength>
</pswd>
```

The parameter maxage defines the number of days a password remains valid. The parameter minage denotes that a password may be changed at most every day. Minlength defines the minimal length of passwords.

The property <historysize>6</historysize> denotes that the last 6 passwords may not be used when changing the password.

You can define a threshold on the number of failed login attempts after which a user account is blocked. This is done using <lockoutthreshold>3</lockoutthreshold>. <lockoutresettimeout>↙ ↳ 60</lockoutresettimeout> specifies the time after wich automatically locked accounts can be used again.

The <characterset> definitions specify which characters for a password are approved and how many characters from a particular group must be used.

```
<characterset>0123456789</characterset>
<choosemin>2</choosemin>
<characterset>abcdefghijklmnopqrstuvwxyz</characterset>
<choosemin>1</choosemin>
<characterset>ABCDEFGHIJKLMNOPQRSTUVWXYZ</characterset>
<choosemin>1</choosemin>
<characterset>!$%+~#@</characterset>
<choosemin>2</choosemin>
```

In the above example, it is specified that from the first and last group (the digits and special characters) at least 2 characters must be used. From the two remaining groups at least a single character must be used.

Note that the specified number denotes a lower bound on the characters chosen from this group.

A complete configuration of the password policy might thus look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <rs>
    <security>
      <passwordpolicy>
        <bsipasswordpolicy>
          <pswd>
            <maxage>32000</maxage>
            <minage>1</minage>
            <minlength>8</minlength>
          </pswd>
          <historysize>6</historysize>
          <lockoutthreshold>3</lockoutthreshold>
          <lockoutresettimeout>60</lockoutresettimeout>
          <characterset>0123456789</characterset>
          <choosemin>1</choosemin>
          <characterset>abcdefghijklmnopqrstuvwxyz</characterset>
          <choosemin>1</choosemin>
          <characterset>ABCDEFGHIJKLMNOPQRSTUVWXYZ</characterset>
          <choosemin>1</choosemin>
          <characterset>!$%&amp;/=?*:.;,-_+~\#@</characterset>
          <choosemin>1</choosemin>
        </bsipasswordpolicy>
      </passwordpolicy>
```

```
    </security>
  </rs>
</configuration>
```

Customizing the Logout URL

By default, when a user logs out, the user will be taken back to the original ReportServer login page. In order to change this, you can change the config file located in etc/security/misc.cf within the ReportServer filesystem.

An example of this is shown below.

```
<logout>
     <url>http://your-reportserver/ReportServer/reportserver/fileServerAccess?path=/↙
         ↳ resources/public/login.html</url>
</logout>
```

The example above redirects the user to the given URL instead of the origianl ReportServer login page.

This is useful when customizing the login page, you can find more details on customizing the login page on Section 4.4 Customization of the Login Page.

Notifications

Users can be notified when their password has been created the first time or when their password was changed (by an administrator). The notification is done via email (note that for this the mail server must be correctly configured). For this purpose, the following configuration file is available: security/notifications.cf.

This configuration file allows to configure the texts sent by email and further allows to disable this functionality, if desired.

```
<createdpassword disabled="false">
  <email>
    <subject>Email Subject</subject>
    <text>Email Text
      Username: ${user.getUsername()}
    </text>
    </email>
</createdpassword>
<changedpassword disabled="false">
  <email>
    <subject>Email Subject</subject>
    <text>Email Text
      Username: ${user.getUsername()}
    </text>
  </email>
</changedpassword>
```

The following substitutions are available.

Expression	Description
${user.getUsername()}	username
${user.getFirstname()}	user's first name
${user.getLastname()}	Last name of user
${user.getEmail()}	user's email address
${user.getTitle()}	user's title
${user.getId()}	user's id

Further, this file allows to configure the texts of the email sent by the `listlogfiles` terminal command using the `-e` option.

```
<logfiles>
  <email>
    <subject>Email Subject</subject>
    <text>Email Text
      Filter: ${filter}
    </text>
  </email>
</logfiles>
```

In this section, the following substitutions are available. For more information, check the `listlogfiles` documentation.

Expression	Description
${filter}	filter passed with -f parameter
${user.getUsername()}	username
${user.getFirstname()}	user's first name
${user.getLastname()}	Last name of user
${user.getEmail()}	user's email address
${user.getTitle()}	user's title
${user.getId()}	user's id

Tip. Note that as of RS 3.3.0 the old lostpassword.cf configuration file is no longer available. This configuration is now done in the new notifications.cf file.

User activation

Users can be activated by administrators using the user manager. On activation, the user will receive an email with an automatically generated (single use) password (note that for this the mail server must be correctly configured). After the first login, the user must change the password according to the password policy.

You can customize the email sent to the user in the configuration file `/fileserver/etc/security/activateuser.cf`.

```
  <security>
    <activateaccount>
      <email>
        <subject>Your ReportServer account details</subject>
```

```
        <text>
          Username: ${user.getUsername()}
          Password: ${password}
        </text>
      </email>
    </activateaccount>
  </security>
```

The following substitutions are available.

Expression	Description
${user.getUsername()}	username
${user.getFirstname()}	user's first name
${user.getLastname()}	Last name of user
${user.getEmail()}	user's email address
${user.getTitle()}	user's title
${user.getId()}	user's id
${password}	the generated password
${url}	the URL under which ReportServer can be accessed

Configuring the SFTP Server

ReportServer Enterprise Edition can be configured to expose its internal filesystem (and other management areas) using an SFTP server. The corresponding configuration goes into `/fileserver/etc/misc/misc.cf`.

```
<remoteaccess>
  <sftp disabled="false">
      <!-- Use $generated in order to generate a key on first start.
        ↪  -->
    <keylocation>/path/to/hostkey.pem</keylocation>
    <port>8022</port>
  </sftp>
</remoteaccess>
```

The SFTP server can be disabled if you don't need it via the `disabled` property. After a ReportServer restart, it will not be started if disabled previously.

The file `hostkey.pem` should contain the server's certificate. You can also use `$generated` ↪ in order to generate a key on first start. The path should be an absolute path (e.g., `file:///C:/path/to/hostkey.pem` in Windows or `/path/to/hostkey.pem` in Unix.) Note that the `file://` protocol is necessary in Windows in order to recognize `C` as the beginning of an absolute path.

Note that changes will only take effect after restarting ReportServer. If you do not want to start the SFTP server simply supply an invalid path or `$generated` and disable it with `disabled`.

4.16 SSO related settings

In the following section we describe ***SSO*** configuration options.

LDAP

LDAP-related properties, described below, are defined in the `/fileserver/etc/sso/ldap.cf` configuration file. These are relevant for the "ldapimport" terminal command and the `ldapimport.groovy` script available here: https://github.com/infofabrik/reportserver-samples/blob/main/src/net/datenwerke/rs/samples/admin/ldap/ldapimport.groovy.

Note that you can (and should) test your LDAP configuration with the `ldaptest` commands described in the Administration Guide for checking your LDAP configuration before letting the real import to happen.

The `ldapschema`, `ldapguid`, `ldapfilter` and `ldapinfo` terminal commands may also be useful for exploring your LDAP server and also the extended the `ldaptest users`, `ldaptest groups` and `ldaptest organizationalUnits` with a `-s` (schema) flag.

While the `-s` flag allows you to explore the installed object class types of your users', OUs' and groups' object classes, the `ldapschema` allows you to explore any object class.

For example, you may execute `ldaptest users -s` for printing the schema of the users' object class. You should get a list of optional attributes, required attributes, and the parent object class. Suppose the parent's object class is "organizationalPerson". You may then explore this object class with `ldapschema objectClassInfo organizationalPerson`.

You may continue exploring the LDAP schemas until the top-most object class: `top`.

```
<disabled>true</disabled>
<provider>
   <host>directory.example.com</host>
   <port>389</port>
</provider>
```

The `disabled` property allows you to completely disable your LDAP if you don't need it. Note that LDAP is disabled by default.

The `provider` property configures the host (or IP) and port where your LDAP server is installed. Note that if you use SSL (LDAPS) this port is different than the LDAP port. StartTLS uses the same LDAP port.

```
<security>
   <encryption>none</encryption>
   <principal>CN=ldaptest,CN=Users,DC=directory,DC=example,DC=com</↙
      ↳ principal>
   <credentials>password</credentials>
</security>
```

The `encryption` property defines the encryption protocol to use. Valid values are `none` (for no encryption), `starttls` (for StartTLS encryption (***recommended***)) and `ssl` (for SSL (LDAPS) encryption).

In order for encryption to work, you have to install the certificates needed for these to be trusted by

ReportServer.

This means that you must add the LDAP server's certificate (or a certificate higher up the trust chain) to a truststore that is known to ReportServer during startup.

This can be achieved in two different ways:

- Passing the truststore where the certificate is installed. If you use JKS keystores you can pass the keystore analogous to:

  -Djavax.net.ssl.trustStore=/path/to/security/truststore.jks

  -Djavax.net.ssl.trustStorePassword=myTrustStorePassword

  -Djavax.net.ssl.trustStoreType=JKS

  If you use PKCS12 you can pass the keystore analogous to:

  -Djavax.net.ssl.trustStore=/path/to/security/truststore.p12

  -Djavax.net.ssl.trustStorePassword=myTrustStorePassword

  -Djavax.net.ssl.trustStoreType=PKCS12

  Refer to the Java Documentation for details:

  https://docs.oracle.com/en/java/javase/11/security/java-secure-socket-extension-jsse-refe
  html

  Be aware that if you use this method, you may need to add other certificates as well in order for your Email, SFTP, OneDrive, etc to continue working, as these certificates are contained in the cacerts truststore, see below. You may of course create a copy of cacerts and add your certificates to this copy instead of using the java cacerts truststore directly.

- Or installing the certificate into your JVM trust store (usually located here `java-home/lib/security/cacerts`). Refer to the Java Documentation for details:

  https://docs.oracle.com/en/java/javase/11/security/java-secure-socket-extension-jsse-refe
  html.

You can test your SSL configuration, i.e. if your certificate was installed correctly, with the `ssltest` terminal command. Check the Administration Guide for details.

The `principal` and `credentials` properties allow you to authenticate to your LDAP server.

```
<base>OU=EXAMPLE,DC=directory,DC=example,DC=com</base>
<filter>
<![CDATA[
(|(objectClass=organizationalUnit)(objectClass=user)(objectClass=group
    ↳ ))
]]>
</filter>
```

The `base` property defines the address of the root object in the LDAP directory. All objects are stored below the base.

The `filter` allows you to retrieve a subset of all the nodes found below the base DN.

You can analyze your installed LDAP filter with the `ldapfilter` terminal command. You can test your installed LDAP filter with the `ldaptest filter` terminal command. Check the Administration Guide for details on both commands.

Note that you have to escape some characters in your filter (e.g. parentheses, space characters, `*`, etc) if you need these in your filter attributes. Example filter:

```
<![CDATA[
(o=Parens\20R\20Us\20\28for\20all\20your\20parenthetical\20needs\29)
]]></filter>
```

In the example you see `\28` instead of `|`, `\29` instead of `)`, and `\20` instead of space . More information can be found here: https://www.rfc-editor.org/rfc/rfc4515#section-4.

If you do not use `CDATA`, you also need to escape `&` with `&`. We recommend to use `CDATA` for better readability.

```
<externalDir>/usermanager/external</externalDir>
<writeProtection>true</writeProtection>
<deleteGroups>true</deleteGroups>
<logResultingTree>false</logResultingTree>
<flattenTree>false</flattenTree>
```

The `externalDir` property defines the directory in ReportServer where your users/groups and OUs will be imported into. These objects will be write-protected if the `writeProtection` is set to `true`.

The `deleteGroups` property controls whether groups deleted in LDAP are also removed from ReportServer. It is recommended to keep this property set to `true` to ensure ReportServer reflects the LDAP structure. If set to `false`, deleted groups in LDAP will remain in ReportServer.

When the `logResultingTree` property is set to `true`, a summary and some statistics of the changes done in your ReportServer are logged into your logs.

If you need to import all nodes into the root directory (i.e. into the `externalDir` directory described above) instead of using the original LDAP tree, you can set `flattenTree` to `true`. Note that all OUs will be empty in this case. If you don't want to include the empty OUs, you have to remove them via the filter attribute.

```
<attributes>
   <objectClass>objectClass</objectClass>
   <guid>entryUUID</guid>
   <organizationalUnit>
      <objectClass>organizationalUnit</objectClass>
      <name>name</name>
   </organizationalUnit>
   <group>
      <objectClass>group</objectClass>
      <name>name</name>
      <member>member</member>
   </group>
```

```
    <user>
       <objectClass>inetOrgPerson</objectClass>
       <firstname>givenName</firstname>
       <lastname>sn</lastname>
       <username>sAMAccountName</username>
       <mail>mail</mail>
    </user>
</attributes>
```

The properties above define the attributes used in your AD nodes. In order to browse and analyze the LDAP schema of your LDAP installation, you can use the following commands: `ldapschema`, `ldapinfo`, and `ldapguid`. Check your Administration Guide for details.

General object class attributes are defined by the `attributes - objectClass` attribute, while object GUIDs are defined by the `guid` attribute. Note that GUIDs must be unique. ReportServer makes a best-effort guess of the appropriate GUID for your installation with the `ldapguid` terminal command. Check your Administration Guide for details.

The `organizationalUnit - objectClass` defines a OU with name specified by `organizationalUnit - name`. The same applies to the `group` attributes.

The `user` attributes specify a given user. The node is determined as a user by the `user - objectClass` attribute, while the `firstname`, `lastname`, `username` and `mail` attributes allow ReportServer to fetch these attributes from a user node.

```
<attributes>
    <additional>
       <attribute>department</attribute>
       <attribute>office</attribute>
    </additional>
</attributes>
```

If you need additional attributes, i.e. attributes not included in the standard attribute list, you can fetch them from your LDAP by defining them in the `additional` list as shown above. The example would fetch the `department` and `office` attributes, which can be then used in a `LdapNodePostProcessHook` hooker as shown in this example: `https://github.com/infofabrik/reportserver-samples/blob/main/src/net/datenwerke/rs/samples/admin/ldap/ldapUserVariableProces groovy`. The example uses the `department` LDAP attribute in order to set the appropriate value into a given user-variable `myUserVar`.

Note that the old `allowLocalUsers` setting was removed as of ReportServer 4.5.0. Instead, you should use colon-separated PAMs in your `reportserver.properties` as explained in the Script Guide (Custom Authenticators PAMs) if you need to allow/disallow local users to log-in.

E.g. the following setting allows only LDAP users:

```
rs.authenticator.pams = net.datenwerke.rs.ldap.service.ldap.pam.LdapPAM
```

while the following setting allows both LDAP and local users:

```
rs.authenticator.pams = net.datenwerke.rs.ldap.service.ldap.pam.LdapPAM:net.datenwerke.rs.a
```

4.17 TeamSpace related settings

The `/etc/security/teamspace.cf` configuration file allows you to configure some TeamSpace settings.

You can completely disable or enable the TeamSpace file upload by the following property:

```
<fileupload disabled = "false">
```

The default is `false`, so the TeamSpace file upload is enabled by default.

Futher, the maximum upload file size can be configured using the following property:

```
<maxSizeBytes>52428800</maxSizeBytes>
```

Default is `50 MB`.

Only the file endings explicitely allowed in the whitelist as shown below are allowed.

```
<whitelist>
   <endings>
      <ending>pdf</ending>
      <ending>jpg</ending>
      <ending>jpeg</ending>
      <ending>png</ending>
      <ending>xlsx</ending>
      <ending>xls</ending>
      <ending>docx</ending>
      <ending>doc</ending>
      <ending>txt</ending>
   </endings>
</whitelist>
```

Note that regular expressions (refer to https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/regex/Pattern.html) are allowed in the ending whitelist. E.g., the following allows all possible file endings:

```
<whitelist>
   <endings>
      <ending>.*</ending>
```

```
    </endings>
</whitelist>
```

Chapter 5

External Configdir

ReportServer provides an alternative mechanism for providing configuration called the ***configdir*** mechanism. This allows to keep system settings separate from application files. You can use the config dir to

- store hibernate connection properties
- override reportserver.properties values
- move the config files from the internal fileserver to you local filesysystem
- import contents into the internal fileserver
- hold additional jars (for example, additional JDBC drivers)
- override logging-rs.properties

To make ReportServer use the configdir you have to set the rs.configdir system property, for example by specifying the

```
-Drs.configdir=/var/lib/rsconfig
```

command line switch.

hibernate connection properties

To use the configdir to specify hibernate connection properties, place a file with the name persistence.properties in the config dir. The file has the standard java properties file format. The properties defined here overwrite the properties configured in the persistence.xml file in the application directory.

For example you could create a persistence.properties file with these contents

```
hibernate.connection.url=jdbc:mysql://localhost:3306/reportserver
hibernate.connection.username=root
hibernate.connection.password=root
```

to provide the base connection parameters for ReportServer to use.

reportserver.properties values

You can override the values defined in the reportserver.properties file in the application directory by creating a second reportserver.properties file in the configdir. Settings from this file will override settings made in the default reportserver.properties file.

additional jars

The lib directory can hold additional jars (for example, additional JDBC drivers). This allows you, in particular, to maintain a set of libraries that will not be overwritten during your next reportserver upgrade.

logging-rs.properties values

You can override the values defined in the logging-rs.properties file in the application directory by creating a second logging-rs.properties file in the configdir. Settings from this file will override settings made in the default logging-rs.properties file.

xml configfiles

You can use the configdir to replace the internal fileserver storage for ReportServer xml configuration files. To use this feature create a config subdirectory in the configdir and place the config files there using the same directory layout as in the fileserver/etc directory. For example to store terminal aliases outside the internal fileserver create a file `{rs.configdir}/config/terminal/alias.cf`. You can use the baseconfig zip file in the pkg subdirectory of you ReportServer download to quickly create the correct directory structure. If a configfile is present in the internal fileserver, the same file on the local filesystem will be ignored, ReportServer will not try to merge the two files. However you can have some config files in the fileserver and others in the configdir.

fileserver import

If you create a fsimport subdirectory in the configdir folder ReportServer will copy its contents to its internal fileserver on startup. This will update/override existing files, however missing files will not be removed.

Creating the configdir directory

In the Linux https://reportserver.net/en/tutorials/installation-best-practice/ and Windows https://reportserver.net/en/tutorials/installation-windows/ Best-Practice Guides you can find examples of how to create the configdir and what files and directories to include in it.

Appendix A

Config File Reference

In our github rs-samples repository you can find the defaults for each configuration file: https://github.com/infofabrik/reportserver-samples/tree/main/config