InfoFabrik gmbh

ReportServer

Configuration Guide 3.2.0

InfoFabrik gmbh

ReportServer

Configuration Guide 3.2.0

InfoFabrik GmbH, 2020

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

Contents

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 various platforms, virtual machines and cloud images are provided by means of the Bitnami ReportServer Stack.

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 Bitnami 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 Bitnami ReportServer Stack is available for Windows, Linux and Mac OS and provides a pre-configured installation of ReportServer with Apache Tomcat as application server and MySQL as database backend. The installers are available from http://reportserver.net/download or directly from Bitnami. 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 MySQL database running and the default ports are in use, you are asked to provide alternative ports. Documentation on the installation via the Bitnami installer is provided by Bitnami and can be accessed from https://bitnami.com/stack/reportserver/README.txt.

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 Oracle Java Runtime Environment (JRE) in version 7 or 8 (Java 8 is supported as of ReportServer 3.0).

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://community.jboss.org/wiki/SupportedDatabases2.

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 PostgrSQL database, for example, use the file:

`reportserverRS2.x-svn-xxx-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 Bitnami 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://community.jboss.org/wiki/SupportedDatabases2

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 *PostgreSQL*

```
# PostgreSQL
hibernate.dialect=org.hibernate.dialect.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=org.hibernate.dialect.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

[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.

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.

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. 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.

`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, i.e. during a first run. This includes the audit log and other data.

`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`.

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 name of the datasource, or use the ID of the datasource, for the assignment. Use one of the two variants for the configuration:

```
<defaultDatasourceName>Myds<defaultDatasourceName>
<defaultDatasourceId>12</defaultDatasourceId>
```

Remark. Keep in mind that the datasource name 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.

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 Data Source</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 name) 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 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.

4.3 Setting up the Scheduler

With ReportServer you can schedule reports and distribute the result either using mail, directly into a ***TeamSpace***, or sent to a FTP server. The next section discusses the relevant settings.

Mail Server Configuration

In order for ReportServer to be able to send mails you must specify the mail server settings. Make the following configurations in the file `/fileserver/etc/mail/mail.cf`.

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

```
<smtp>
  <host>mail.yourmailserver.com</host>
  <port>25</port>
  <username>rs@yourmailserver.com</username>
  <password>passwordsecret</password>
  <ssl>false</ssl>
  <tls>
    <enable>false</enable>
    <require>false</require>
  </tls>
</smtp>
```

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.

```
<mail>
  <sender>rs@yourmailserver.com</sender>
  <senderName>ReportServer</senderName>
  <forceSender>false</forceSender>
  <encryptionPolicy>allow_mixed</encryptionPolicy>
</mail>
```

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.

FTP Server Configuration

In order for ReportServer to be able to send reports via FTP you must specify the FTP server settings. Make the following configurations in the file `/fileserver/etc/exportfilemd/storage.cf`.

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

```
<ftp disabled="false" supportsScheduling="true">
  <host>ftp.host.net</host>
  <port>21</port>
  <username>rs@host.net</username>
  <password></password>
  <defaultFolder>./</defaultFolder>
</ftp>
```

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

The `defaultForder` allows you to specify where exactly the reports should be saved by default in the FTP server. This can be overwritten by the specific scheduler job.

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 emailed, stored in a folder in a ***TeamSpace***, or sent to a FTP server.

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 with attached completed report (mailaction),
- email if a report has been placed into a TeamSpace (fileaction),
- email if a report has been placed into a FTP server (fileactionFtp),
- 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
`${report.getName()}`	report's name
`${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 and storage in FTP server

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

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">
```

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/schedule` 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.

4.4 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>
```

You can specifiy 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 specifiy 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.

4.5 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,
- Customize the theme,
- customize PDF preview,
- customize the report documentation,
- add 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 error messages

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.

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 two parts:

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

  </adminviews>
  <objectinfo>

  </objectinfo>
```

```
</configuration>
```

Tabs to be displayed in the administration module go into the **adminviews** tags and tabs for the TeamSpace are put within the **objectinfo** tags, respectively. The default configuration does not add any additional tabs for the admin interface, 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>
</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.

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 following tables 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>
```

4.6 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 (groovy.codehaus.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.rs</login>
    <rs>fileserver/bin/onstartup.rs</rs>
  </startup>
</scripting>
```

In the above example we allow scripts only in the bin folder (and subfolders). We defined the script `fileserver/bin/onlogin.rs` 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.rs` 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***.

4.7 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.

4.8 Misc Settings

ReportServer needs a directory to store temporary files. You can define the directory to use in `/filserver/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>
```

4.9 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.

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>!$%&amp;/=?*:.;,-\_+~#@</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>
```

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.c`

```
<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.

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

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.

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.

Appendix A

Config File Reference

In the following Chapter we provide samples for each configuration file.

External Configuration Files

persistence.properties

reportserver.properties

Internal Configuration Files

datasources/databasebundle.cf

datasources/datasources.cf

datasources/internaldb.cf

datasources/parameter.cf

datasources/pool.cf

datasources/sql.cf

dynamiclists/computedcolumn.cf

dynamiclists/htmlexport.cf

dynamiclists/pdfexport.cf

exportfilemd/excelexport.cf

exportfilemd/metadata.cf

exportfilemd/pdfexport.cf

mail/mail.cf

A.1 External Configuration Files

persistence.properties

```
#
# file: persistence.properties
# description: This file contains the database settings for ReportServer

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

# Credentials
# hibernate.connection.username=root
# hibernate.connection.password=root


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


# 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
```

```
# PostgreSQL
# hibernate.dialect=net.datenwerke.rs.utils.hibernate.PostgreSQLDialect
# hibernate.connection.driver_class=org.postgresql.Driver
# hibernate.connection.url=jdbc:postgresql://localhost/reportserver


# 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:reportserver
# hibernate.default_schema=


# 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;databaseName=reportserver


# DB2
# hibernate.dialect=net.datenwerke.rs.utils.hibernate.DB2Dialect
# hibernate.connection.driver_class=com.ibm.db2.jcc.DB2Driver
# hibernate.connection.url=jdbc:db2://localhost:50000/TEST


# Connection pool 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.
#
# If no changes are made here, then the default settings are active
#
# 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
```

reportserver.properties

```
### reportserver.properties ######################################################
# This file controls general reportserver parameters.
#
# NOTE:
# Most of the properties formerly configured in this file where migrated
# over to the etc tree of ReportServers internal filesystem.
#
################################################################################



### crypto properties ############################################################

  # rs.crypto.pbe.salt
  # configures the salt used in password based encryption
  # reportserver uses password based encryption to store database password and
```

```
  # other sensitive information, that can not be stored as a hash value
    rs.crypto.pbe.salt = The salt to be used for encryption. This should simply be a ↙
        ↳ long string.

  # rs.crypto.pbe.passphrase
  # the passphrase used in password based encryption
    rs.crypto.pbe.passphrase = The Passphrase

  # rs.crypto.pbe.keylength = 128
  # the maximum keylength used with password based encryption
    rs.crypto.pbe.keylength = 128

  # rs.crypto.passwordhasher.hmac.passphrase
  # the passphrase used when calculating hmac
     rs.crypto.passwordhasher.hmac.passphrase = This is the Passphrase used to compute ↙
         ↳ the HMAC key for reportServer passwords.

### authenticator configuration ################################################

  # rs.authenticator.pams
  # configures the pluggable modules the authenticator uses to verify requests
  # multiple modules are separated by colon ":" characters
  # possible values are:
  #   net.datenwerke.rs.authenticator.service.pam.UserPasswordPAM
  #   net.datenwerke.rs.authenticator.service.pam.UserPasswordPAMAuthoritative
  #   net.datenwerke.rs.authenticator.service.pam.IPRestrictionPAM
  #   net.datenwerke.rs.authenticator.service.pam.EveryoneIsRootPAM
  #   net.datenwerke.rs.authenticator.cr.service.pam.ChallengeResponsePAM
  #   net.datenwerke.rs.authenticator.cr.service.pam.ChallengeResponsePAMAuthoritative
  #   net.datenwerke.rs.authenticator.service.pam.ClientCertificateMatchEmailPAM
  #   net.datenwerke.rs.authenticator.service.pam.↙
      ↳ ClientCertificateMatchEmailPAMAuthoritative
    rs.authenticator.pams = net.datenwerke.rs.authenticator.service.pam.↙
        ↳ UserPasswordPAMAuthoritative

  # rs.authenticator.iprestriction.addresses
  # if the IPRestrictionPAM is active, this property controls the acceptable
  # soruce addresses
  #
  # rs.authenticator.iprestriction.addresses = 127.0.0.1/32:192.168.1.0/24

  # rs.authenticator.pam.ClientCertificateMatchEmailPAM.debug
  # this property enables debug output for the ClientCertificateMatchEmailPAM
    rs.authenticator.pam.ClientCertificateMatchEmailPAM.debug = false

  # rs.authenticator.blockroot
  # this property disables all accounts with root privileges
    rs.authenticator.blockroot = false


### general settings ###########################################################

  # rs.install.basedata = true
  # creates basedata if database is empty
    rs.install.basedata = true

  # rs.scripting.disable = true
  # this property disables scripting

  # rs.scheduler.disable = true
  # this property disables the scheduler
```

A.2 Internal Configuration Files

datasources/databasebundle.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/databasebundle.cf

 Configures available datasource keys in datasource bundles
-->
<configuration>
   <!--
   <bundletype>
      <name>BaseBundle</name>
      <keys>
         <key>Production System</key>
         <key>Test System</key>
      </keys>
   </bundletype>
   -->
</configuration>
```

datasources/datasources.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/datasources.cf

 Allows to configure a default datasource for easy access.
-->
<configuration>
   <datasource>
      <defaultDatasourceName>Demo Data</defaultDatasourceName>
      <!-- or access via ID -->
      <!-- <defaultDatasource>14</defaultDatasource> -->
   </datasource>
</configuration>
```

datasources/internaldb.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/internaldb.cf

 Configures the internaldb used by ReportServer for non-relational datasources.
-->
<configuration>
   <internaldb>
      <droponstartup>true</droponstartup>
      <datasource>ReportServer Data Source</datasource>
   </internaldb>
</configuration>
```

datasources/parameter.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/parameter.cf

 Provides settings for datasource and date time parameters.
-->
<configuration>
```

```
   <parameter>
      <!-- settings for datasource parameters -->
      <datasource>
         <!-- specify a time out for database queries to parameters -->
         <querytimeout>60</querytimeout>

         <!-- enable or disable post processing of parameter queries -->
         <postprocessing>
            <enable>true</enable>
         </postprocessing>
      </datasource>

      <!-- settings for date time parameters -->
      <date>
        <urlpattern></urlpattern>
      </date>
   </parameter>
</configuration>
```

datasources/pool.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/pool.cf

 Allows to configure the connection pool settings for database datasources.
-->
<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>
   </pool>
</configuration>
```

datasources/sql.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: datasources/datasources.cf

 Configures the internal SQL-query builder.
-->
<configuration>
   <sql>
      <!-- define a maximum number of records that can go into an IN clause -->
      <incondition>
         <maxsize>1000</maxsize>
      </incondition>
      <quoteIdentifiers>smart</quoteIdentifiers>
   </sql>
</configuration>
```

dynamiclists/computedcolumn.cf

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

```
 ReportServer Configuration File
 filename: dynamiclists/computedcolumn.cf

 Provides a white-list of functions that can be used for computed columns in dynamic ↙
     ↳ lists.
-->
<configuration>
   <functions>
      <function>abs</function>
      <function>ceil</function>
      <function>floor</function>
      <function>mod</function>
      <function>power</function>
      <function>round</function>
      <function>sign</function>
      <function>sin</function>
      <function>sqrt</function>
      <function>trunc</function>
      <function>exp</function>
      <function>ln</function>
      <function>log</function>
      <function>concat</function>
      <function>lower</function>
      <function>upper</function>
      <function>initcap</function>
      <function>lpad</function>
      <function>rpad</function>
      <function>ltrim</function>
      <function>rtrim</function>
      <function>trim</function>
      <function>replace</function>
      <function>translate</function>
      <function>substr</function>
      <function>instr</function>
      <function>length</function>
      <function>add_months</function>
      <function>last_day</function>
      <function>months_between</function>
      <function>round</function>
      <function>sysdate</function>
      <function>trunc</function>
      <function>to_char</function>
      <function>to_date</function>
      <function>to_number</function>
      <function>hextoraw</function>
      <function>rawtohexcomput</function>
   </functions>
</configuration>
```

dynamiclists/htmlexport.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: dynamiclists/htmlexport.cf

 Styles the HTML export of dynamic lists
 This file is only active in ReportServer Enterprise Edition
-->
<configuration>
   <htmlexport>
       <!--
        <title>Some title</title>
        <head>Some additional content for the header</head>
```

```
        <script>Some Javascript</script>
       -->
      <style><![CDATA[
         @page {
          size: A4 landscape;
          @top-left {
            content: "${report.name}";
            font-family: DejaVu Sans, Sans-Serif;
            font-size: 8pt;
          }
            @top-right {
            content: "${now}";
                font-family: DejaVu Sans, Sans-Serif;
                font-size: 8pt;
            }
            @bottom-right {
            content: "${page} " counter(page) " ${of} " counter(pages);
              font-family: DejaVu Sans, Sans-Serif;
               font-size: 8pt;
            }
          }
          ]]>
      </style>
      <pre><![CDATA[
<div class="wrap">
<div class="header">
<span class="logo">
<img src="SomeLogoUrl" alt="logo"></img>
</span>
<div class="reportdata">
<span class="name">${report.name}</span>
<span class="date">${now}</span>
</div>
<div class="clear"></div>
</div>

<!-- output parameters / filters before report data -->
<!-- Activate per report with output_parameters, output_filters, or ↙
    ↳ output_complete_configuration report property.
  The property has to be set to true in order to activate. -->
<!-- <div class="parameters">
You can export parameters with the parameterMapSimple variable.
</div>
<div class="filters">
You can export filters with the filterMapSimple variable.
</div> -->
<!-- end output parameters / filters before report data -->

]]>
      </pre>
      <post><![CDATA[

<!-- output parameters / filters after report data -->
<!-- Activate per report with output_parameters, output_filters, or ↙
    ↳ output_complete_configuration report property.
  The property has to be set to true in order to activate. -->
<!-- <div class="parameters">
You can export parameters with the parameterMapSimple variable.
</div>
<div class="filters">
You can export filters with the filterMapSimple variable.
</div> -->
<!-- end output parameters / filters after report data -->
```

```
</div>
]]></post>
   </htmlexport>
</configuration>
```

dynamiclists/pdfexport.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: dynamiclists/pdfexport.cf

 Styles the PDF export of dynamic lists
 This file is only active in ReportServer Enterprise Edition
-->
<configuration>
   <htmlexport>
      <style><![CDATA[
         @page {
          size: A4 landscape;
          @top-left {
            content: "${report.name}";
            font-family: DejaVu Sans, Sans-Serif;
            font-size: 8pt;
          }
            @top-right {
            content: "${now}";
                font-family: DejaVu Sans, Sans-Serif;
                font-size: 8pt;
            }
            @bottom-right {
            content: "${page} " counter(page) " ${of} " counter(pages);
              font-family: DejaVu Sans, Sans-Serif;
               font-size: 8pt;
            }
          }
          * {
        font-family: DejaVu Sans, Sans-Serif;
      }
          ]]>
      </style>
      <pre><![CDATA[
<div class="wrap">
<div class="header">
<span class="logo">
<img src="SomeLogoUrl" alt="logo"></img>
</span>
<div class="reportdata">
<span class="name">${report.name}</span>
<span class="date">${now}</span>
</div>
<div class="clear"></div>
</div>

<!-- output parameters / filters before report data -->
<!-- Activate per report with output_parameters, output_filters, or ↙
    ↳ output_complete_configuration report property.
  The property has to be set to true in order to activate. -->
<!-- <div class="parameters">
You can export parameters with the parameterMapSimple variable.
</div>
<div class="filters">
You can export filters with the filterMapSimple variable.
</div> -->
```

```
<!-- end output parameters / filters before report data -->

]]>
      </pre>
      <post><![CDATA[

<!-- output parameters / filters after report data -->
<!-- Activate per report with output_parameters, output_filters, or ↙
    ↳ output_complete_configuration report property.
  The property has to be set to true in order to activate. -->
<!-- <div class="parameters">
You can export parameters with the parameterMapSimple variable.
</div>
<div class="filters">
You can export filters with the filterMapSimple variable.
</div> -->
<!-- end output parameters / filters after report data -->

</div>
]]></post>
   </htmlexport>
</configuration>
```

exportfilemd/excelexport.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <xls>
      <format>xlsx</format>
      <stream>true</stream>
   </xls>
</configuration>
```

exportfilemd/metadata.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <exportmetadata>
      <title>Title</title>
      <creator>ReportServer</creator>
      <author>ReportServer</author>
   </exportmetadata>
</configuration>
```

exportfilemd/pdfexport.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <pdf>
   <!--
      <fonts>
        <font>
          <path></path>
          <encoding></encoding>
          <embed></embed>
      </font>
      </fonts>
   -->
   </pdf>
</configuration>
```

mail/mail.cf

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

```
      <host>mail.host.net</host>
      <port>25</port>
      <!--<username>rs@host.net</username>
      <password></password>
      -->
      <ssl>false</ssl>
      <tls>
         <enable>false</enable>
         <require>false</require>
      </tls>
   </smtp>
   <mail>
      <sender>rs@host.net</sender>
      <senderName>ReportServer</senderName>
      <forceSender>false</forceSender>
      <encryptionPolicy>allow_mixed</encryptionPolicy>
   </mail>
</configuration>
```

main/localization.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <localization>
      <default>en</default>
      <!-- <locales>en,fr,de</locales> -->
      <format>
      <!--
         <shortDatePattern></shortDatePattern>
       <longDatePattern></longDatePattern>
       <shortTimePattern></shortTimePattern>
       <longTimePattern></longTimePattern>
       <shortDateTimePattern></shortDateTimePattern>
       <longDateTimePattern></longDateTimePattern>
       <numberPattern></numberPattern>
       <currencyPattern></currencyPattern>
       <integerPattern></integerPattern>
       <percentPattern></percentPattern>
    -->
      </format>
   </localization>
</configuration>
```

main/main.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <install>
      <basedata>true</basedata>
   </install>
   <default>
      <charset>UTF-8</charset>
   </default>
   <tempdir>tempdir</tempdir>
   <tempfile>
      <lifetime>3600</lifetime>
   </tempfile>
   <maintenance>
      <tasks>
         <interval>600000</interval>
      </tasks>
   </maintenance>
   <search>
      <timeout>5000</timeout>
   </search>
```

```
   <pagetitle>ReportServer</pagetitle>
</configuration>
```

main/templates.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <errors>
      <html>&lt;!DOCTYPE html PUBLIC &quot;-//W3C//DTD HTML 4.01//EN&quot; &quot;http://↙
          ↳ www.w3.org/TR/html4/strict.dtd&quot;&gt;
&lt;html&gt;
&lt;meta http-equiv=&quot;content-type&quot; content=&quot;text/html; charset=UTF-8&quot↙
    ↳ ;&gt;
&lt;head&gt;
&lt;title&gt;${msgs['net.datenwerke.rs.core.client.locale.ReportServerMessages']['↙
    ↳ htmlErrorPageTitle']}&lt;/title&gt;
&lt;link rel=&quot;stylesheet&quot; type=&quot;text/css&quot; href=&quot;${baseurl}/↙
    ↳ rstheme&quot; /&gt;
&lt;link rel=&quot;stylesheet&quot; type=&quot;text/css&quot; href=&quot;${baseurl}/../↙
    ↳ resources/rs-logo-font/style.css&quot; /&gt;
&lt;/head&gt;
&lt;body class=&quot;rs-errorpage&quot;&gt;
&lt;i class=&quot;icon-rs-logo&quot;&gt;&lt;/i&gt;
&lt;div class=&quot;rs-errorpage-header&quot;&gt;${msgs['net.datenwerke.rs.core.client.↙
    ↳ locale.ReportServerMessages']['htmlErrorPageHeading']} ${headline}&lt;/div&gt;
&lt;div class=&quot;rs-errorpage-msg&quot;&gt;${msg}&lt;/div&gt;
&lt;div class=&quot;rs-errorpage-todo&quot;&gt;${msgs['net.datenwerke.rs.core.client.↙
    ↳ locale.ReportServerMessages']['htmlErrorInstructions']}&lt;/div&gt;
&lt;div class=&quot;rs-errorpage-stacktrace-header&quot;&gt;${msgs['net.datenwerke.rs.↙
    ↳ core.client.locale.ReportServerMessages']['htmlErrorPageDetailSectionHeader']}&lt↙
    ↳ ;/div&gt;
&lt;div class=&quot;rs-errorpage-stacktrace&quot;&gt;${stacktrace}&lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;</html>
   </errors>
</configuration>
```

misc/misc.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <database>
      <oracle>
         <filter>
            <convertclobs>true</convertclobs>
         </filter>
      </oracle>
   </database>
   <remoteaccess>
      <sftp disabled="false">
         <!-- Use $generated in order to generate a key on first start. -->
         <keylocation>$generated</keylocation>
         <port>8022</port>
      </sftp>
   </remoteaccess>
</configuration>
```

reportengines/reportengines.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <birt>
      <library>
         <folder>
            <id>1</id>
```

```
        </folder>
      </library>
      <enable>true</enable>
   </birt>
   <jasper>
      <enable>true</enable>
      <allowedlanguages>groovy</allowedlanguages>
   </jasper>
</configuration>
```

scheduler/scheduler.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <scheduler>
      <mailaction html="false">
         <subject>ReportServer: ${subject}</subject>
         <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
             ↳ SchedulerMessages']['mailactionMsgText']}

  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelReport']} ${report.getName()}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelMessage']} ${message}
      </text>
         <attachment>
            <name>rep-${report.getName()}-${RS_CURRENT_DATE}</name>
         </attachment>
      </mailaction>
      <fileaction disabled="false" html="false">
         <subject>ReportServer: ${msgs['net.datenwerke.rs.scheduler.client.scheduler.↙
             ↳ locale.SchedulerMessages']['fileactionMsgSubject']}</subject>
         <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
             ↳ SchedulerMessages']['fileactionMsgText']}

  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelReport']} ${report.getName()}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelFilename']} ${filename}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelDescription']} ${description}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelTeamspace']} ${teamspace.getName()}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelFolder']} ${folder.getName()}
      </text>
      </fileaction>
      <fileactionFtp disabled="false" html="false">
         <subject>ReportServer: ${msgs['net.datenwerke.rs.scheduler.client.scheduler.↙
             ↳ locale.SchedulerMessages']['fileactionFtpMsgSubject']}</subject>
         <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
             ↳ SchedulerMessages']['fileactionFtpMsgText']}

  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelReport']} ${report.getName()}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelFilename']} ${filename}
  ${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
      ↳ labelFolder']} ${folder}
      </text>
      </fileactionFtp>
      <notification disabled="false" html="false">
         <scheduled>
            <subject>ReportServer: ${msgs['net.datenwerke.rs.scheduler.client.scheduler.↙
```

```
                ↪ locale.SchedulerMessages']['notificationMsgScheduledSubject']}</↙
                ↪ subject>
            <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
                ↪ SchedulerMessages']['notificationMsgScheduledText']}

${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelReport']} ${report.getName()}
${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelUser']} ${scheduleUser}
${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ notificationMsgScheduledNextDates']} ${nextDates}
    </text>
         </scheduled>
         <unscheduled>
            <subject>ReportServer: ${msgs['net.datenwerke.rs.scheduler.client.scheduler.↙
                ↪ locale.SchedulerMessages']['notificationMsgUnscheduledSubject']}</↙
                ↪ subject>
            <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
                ↪ SchedulerMessages']['notificationMsgUnscheduledText']}

${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelReport']} ${reportName}</text>
         </unscheduled>
         <failed>
            <subject>ReportServer: ${msgs['net.datenwerke.rs.scheduler.client.scheduler.↙
                ↪ locale.SchedulerMessages']['notificationMsgFailedSubject']}</subject>
            <text>${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.↙
                ↪ SchedulerMessages']['notificationMsgFailedText']}

${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelReport']} ${report.getName()}
${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelErrorMessage']} ${errMsg}
${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelRecipient']} ${recipients}
${msgs['net.datenwerke.rs.scheduler.client.scheduler.locale.SchedulerMessages']['↙
    ↪ labelErrorDetails']} ${stacktrace}
    </text>
         </failed>
      </notification>
      <properties>
         <disabled>false</disabled>
      </properties>
  </scheduler>
</configuration>
```

scripting/scripting.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: scripting/scripting.cf

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

```
   </scripting>
</configuration>
```

security/activateuser.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <security>
      <activateaccount>
         <email>
            <subject>${msgs['net.datenwerke.rs.passwordpolicy.service.locale.↙
                ↳ PasswordPolicyMessages']['activateEmailSubject']}</subject>
            <text>${msgs['net.datenwerke.rs.passwordpolicy.service.locale.↙
                ↳ PasswordPolicyMessages']['activateEmailSalutation']} ${user.↙
                ↳ getFirstname()} ${user.getLastname()}, &#xD;
&#xD;
${msgs['net.datenwerke.rs.passwordpolicy.service.locale.PasswordPolicyMessages']['↙
    ↳ activateEmailIntro']}
&#xD;
   ${url}&#xD;
&#xD;
${msgs['net.datenwerke.rs.passwordpolicy.service.locale.PasswordPolicyMessages']['↙
    ↳ activateEmailAccount']}
&#xD;
   ${msgs['net.datenwerke.rs.passwordpolicy.service.locale.PasswordPolicyMessages']['↙
       ↳ activateEmailUsername']} ${user.getUsername()}&#xD;
   ${msgs['net.datenwerke.rs.passwordpolicy.service.locale.PasswordPolicyMessages']['↙
       ↳ activateEmailPassword']} ${password}&#xD;
&#xD;
${msgs['net.datenwerke.rs.passwordpolicy.service.locale.PasswordPolicyMessages']['↙
    ↳ activateEmailEnd']}
&#xD;
</text>
         </email>
      </activateaccount>
   </security>
</configuration>
```

security/crypto.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <cryptocredentials>
     <!--
      <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>
      <provider type="sftp">
         <class>net.datenwerke.rs.incubator.service.crypto.↙
             ↳ FileServerKeyStoreKryptoCredentialProvider</class>
         <alias/>
         <secret/>
         <type/>
         <location/>
      </provider>
      <provider type="user">
         <class>net.datenwerke.rs.incubator.service.crypto.↙
             ↳ FileServerKeyStoreKryptoCredentialProvider</class>
         <alias>rs</alias>
         <secret>secret</secret>
```

```
         <type>jks</type>
         <location>/fileserver/keystore-usr.jks</location>
      </provider>
      -->
  </cryptocredentials>
   <pbe>
      <salt>The salt to be used for encryption. This should simply be a long string.</salt>
      <passphrase>The Passphrase</passphrase>
      <keylength>128</keylength>
   </pbe>
   <hmac>
      <passphrase>This is the Passphrase used to compute the HMAC key for reportServer passwords.</passphrase>
   </hmac>
</configuration>
```

security/misc.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <juel>
      <expression>
         <blacklist>getClass</blacklist>
      </expression>
   </juel>
</configuration>
```

security/passwordpolicy.cf

```
<?xml version="1.0" encoding="UTF-8"?>
  <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>!$%&;/=?*:.;,-\_+~#@</characterset>
          <choosemin>1</choosemin>
        </bsipasswordpolicy>
      </passwordpolicy>
    </security>
</rs>
```

terminal/alias.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: terminal/alias.cf

 Allows to provide alias for commands on the terminal.
-->
```

```
<configuration>
   <cmdaliases>
      <entry>
         <alias>ll</alias>
         <command>ls -l</command>
      </entry>
   </cmdaliases>
</configuration>
```

ui/previews.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: ui/previews.cf

 Configures how previews are rendered
-->
<configuration>
   <pdf>
      <!-- possible values are:
        native: to use the browsers native rendering engine
        jsviewer: to use a javascript based engine
        image: to not render PDFs but only provide the first page as an image
        -->
      <mode>native</mode>
   </pdf>
   <dynamicList>
      <defaultColumnWidth>200</defaultColumnWidth>
      <maxColumnWidth>800</maxColumnWidth>
   </dynamicList>
</configuration>
```

ui/theme.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: ui/theme.cf

 Allows to adapt the look and feel of ReportServer
 This file is only active in ReportServer Enterprise Edition
-->
<configuration>
  <theme type="default">
    <header>
      <height>40</height>
    </header>
    <logo>
      <login>
        <html><![CDATA[<i class="icon-rs-logo rs-login-logo"></i><span class="rs-login-↙
            ↳ bg"><i class="icon-rs-logo-square"></i></span>]]></html>
        <width>200px</width>
      </login>
      <header>
          <html><![CDATA[<span class="rs-header-logo"><i class="icon-rs-Report"></i><i ↙
              ↳ class="icon-rs-Server"></i></span>]]></html>
        <width>185px</width>
      </header>
      <!--<report>Some URI pointing to a Logo for use in internal reporting.</report> --↙
          ↳ >
    </logo>

    <colors>
      <color name="white" color="#FFFFFF"/>
```

```
      <color name="black" color="#000000"/>
      <color name="black-almost" color="#132834"/>

      <color name="purple-dark" color="#3E4059"/>
      <color name="purple-light" color="#DFE0EB"/>

      <color name="gray-light" color="#EEEEEE"/>
      <color name="gray-dark" color="#B8BDC0"/>
      <color name="gray-very-dark" color="#6D708B"/>

      <color name="terminal-green" color="#00B000"/>
    </colors>

    <colorMapping>
      <map useFor="bg" colorRef="gray-dark"/>
      <map useFor="bg.text" colorRef="black"/>

      <map useFor="bg.light" colorRef="white"/>
      <map useFor="light.text" colorRef="black"/>

      <map useFor="bg.shaded" colorRef="gray-light"/>
      <map useFor="shaded.text" color="#666666"/>

      <map useFor="hl.dark.bg" colorRef="purple-dark"/>
      <map useFor="hl.dark.text" colorRef="white"/>

    <map useFor="hl.light.bg" colorRef="purple-dark"/>
      <map useFor="hl.light.text" colorRef="purple-light"/>

      <map useFor="header.bg" colorRef="black-almost"/>
      <map useFor="header.text.active" colorRef="white"/>
      <map useFor="header.text.inactive" color="#BBBBBB"/>
    <map useFor="header.text.right" color="#BBBBBB"/>

      <map useFor="terminal.bg" colorRef="black"/>
      <map useFor="terminal.text" colorRef="terminal-green"/>
      <map useFor="terminal.hl.bg" colorRef="gray-very-dark"/>
    <map useFor="terminal.link" colorRef="white"/>

      <map useFor="border.light" colorRef="gray-dark"/>

      <map useFor="tbar.btn.bg" colorRef="gray-dark"/>

      <map useFor="icon.light" color="#999999"/>

    </colorMapping>

    <css>
      .icon-rs-Report {
        color: #FFF !important;
      }
    </css>
  </theme>
</configuration>
```

ui/ui.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: ui/ui.cf

-->
<configuration>
```

```
  <contexthelp enable="false"/>
  <warnonexit>true</warnonexit>
  <popupDownloadMethod>false</popupDownloadMethod>
  <exportReportsInline>false</exportReportsInline>
  <mandatoryParameterPrefix></mandatoryParameterPrefix>
  <mandatoryParameterSuffix></mandatoryParameterSuffix>
  <optionalParameterPrefix></optionalParameterPrefix>
  <optionalParameterSuffix></optionalParameterSuffix>
</configuration>
```

ui/urlview.cf

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 ReportServer Configuration File
 filename: ui/urlview.cf

-->
<configuration>
   <adminviews>
      <!--<view>
         <types>net.datenwerke.rs.core.client.reportmanager.dto.reports.ReportDto/types>
         <name>Some URL in a Tab</name>
     <url>/reportserver/foobar/id=${id}&amp;type=${type}</url>
   </view>-->
   </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>
</configuration>
```