@enterprise *10.0*

Installation and Configuration

December 2018

Groiss Informatics GmbH

Groiss Informatics GmbH

Strutzmannstraße 10/4
9020 Klagenfurt
Austria

Tel: +43 463 504694 - 0
Fax: +43 463 504594 - 10
Email: support@groiss.com

Document Version 10.0.25071

Contents

1 System Requirements

1.1 Platform

@enterprise is available for several platforms. For the operation of a server, a Java Runtime Environment (JRE) of Version 1.8 or higher is required. The following operating systems are supported:

- Windows Variants (2008, Win7, Win8, Win8.1, Win10)
- Solaris
- AIX
- Linux

The server should have at least 512MB of memory for **@enterprise** and 200MB free disk space.

1.2 Java

To develop **@enterprise** applications, a Java Development Kit (JDK) version 1.8 or higher must be installed. It is available for download from the Oracle web site (http://java.oracle.com) or from another vendor. At the Oracle web site, a list of Java ports to other platforms is available.

1.3 Database Management Systems

We support the following DBMSs: Oracle, MS SQL Server, IBM's DB2, PostgreSQL, Derby. MySQL and Firebird is supported experimentally.
The following database versions are required:

- Oracle 9i or higher
- MS SQL Server 2008 or higher
- DB2 9.7 or higher on Windows or AIX

- PostgreSQL V8.4 or higher
- Derby 10.5.3.0 or higher
- MySQL 5.0 (experimental)
- Firebird Version 2.5 or higher (experimental)

The database can be installed on the same machine as **@enterprise** or on another networked server.

1.4 Client

In order to use the the Web-Client, a Web-Browser is all that is needed. Supported products and versions are:

- MS Internet Explorer 11 or higher
- Firefox 44.0 or higher
- Safari 11.0 or higher
- Chrome 48 or higher

2 Installation

2.1 Database Preparation

@enterprise needs a database with one user. In the following we briefly describe the necessary steps for creating a database user for the supported databases.
Please consult the database manuals or the local experts for further information about database setup and creation of a user.

2.1.1 Oracle

You need a database user with the following rights:

```
create session
alter session
create table
create view
```

The user must also have access to a *tablespace* and the permission to add data there.
Example (*EP_USER* is the name of the **@enterprise** database user):

```
create user <EP_USER> identified by <password> default tablespace users;
grant create session, alter session to <EP_USER>;
grant create table, create view to <EP_USER>;
grant unlimited tablespace to <EP_USER>;
```

Since Oracle 11g, a default profile mechanism with resource limitations and password expiration settings might lead to immediate lockout when getting the password wrong or to lockout after a password expiration intervall. It is recommended to check the applicable profile parameters and to change them appropriately for the **@enterprise** database user. An unlimited profile can be created with:

```
create profile <EP_UNLIMITED_PROFILE> limit
  composite_limit unlimited
  connect_time unlimited
  cpu_per_call unlimited
  cpu_per_session unlimited
  failed_login_attempts unlimited
  idle_time unlimited
```

```
logical_reads_per_call unlimited
logical_reads_per_session unlimited
password_grace_time unlimited
password_life_time unlimited
password_lock_time 1
password_reuse_max unlimited
password_reuse_time unlimited
password_verify_function null
private_sga unlimited
sessions_per_user unlimited;
```

The specific requirements for your site may vary, in case of doubt check with your local DBA. The profile can be assigned to the user with:

```
alter user <EP_USER> profile <EP_UNLIMITED_PROFILE>;
```

Other useful commands for account administration and trouble shooting are:

- *Check the users profile*:

  ```
  select profile from dba_users
  where username ='<EP_USER>';
  ```

- *Check the properties of the profile*:

  ```
  select resource_name, limit from dba_profiles
  where profile=
    (select profile from dba_users
     where username ='<EP_USER>');
  ```

- *Check the users account state:*

  ```
  select username,profile,account_status,expiry_date from dba_users
  where username='<EP_USER>';
  ```

- *Unlock a users account:*

  ```
  alter user <EP_USER> account unlock;
  ```

- *Unexpire an account or change a users password:*

  ```
  alter user <EP_USER> identified by <password>;
  ```

Hint: If you got the message *Could not get Session ID. Probably no right on V$SESSION*, you have to carry out the following steps in Oracle:

1. *Login as sys*: sqlplus sys as sysdba
2. *Assign grant*: grant select on v_$session to <EP_USER>;

If you use full-text search the *IndexRefreshTimer* needs an additional right:

```
grant execute on ctxsys.ctx_ddl to <EP_USER>;
```

If the use of full-text search or WfXML2 functionality is intended with Oracle as the underlying DBMS, you must select the Oracle LOBs database type in the configuration (and not the legacy mode with Oracle LONGs). Since Oracle supports just one LONG column per table, the tables for WfXML2 functionality will not be generated when LONGs are used instead of LOBs.

Oracle offers the possibility to set the semantic of varchar/varchar2 datatypes (BYTE or CHAR). The decision of setting the correct type could be necessary, if UTF-8 texts should be stored. An example could be that a field has a length-restriction of 100 characters and the text to be stored contains 100 characters with 2 umlauts. Because of UTF-8 encoding the text will grow up to 102 Byte and could not be stored anymore.
For this purpose you can change the semantic on two ways:

- global by using following statement:

  ```
  alter system set nls_length_semantics='CHAR' scope=both;
  ```

- per session (db-session-environments in **@enterprise** configuration) by using following statement:

  ```
  alter session set nls_length_semantics='CHAR';
  ```

Hints for the performance of Oracle-based **@enterprise** installations can be found in appendix A.

2.1.2 MS SQL-Server

@enterprise requires a case insensitive installation of MS SQL-Server.
When creating a SQL-Server database, use the option 'ANSI NULL is default'. You can specify it in the database property panel or by execution of a stored procedure after installation.

```
sp_dboption <DBNAME>,'ANSI null default', true
```

<DBNAME> must be replaced with the name of your database. The procedure results in behavior consistent with the ANSI standard regarding the handling of NULL values.
The database user for **@enterprise** must have the right to create tables, for example via the role db_owner.

It is advisable to use Statement-Level Snapshot Isolation in order to avoid shared locks by readers. Enable it with:

```
alter database <DBNAME> set read_committed_snapshot on;
```

Note that no other users are permitted to access the database when you issue this command and that the feature is available only in SQLSserver 2005 or higher.
If you use full-text search, please ensure that MSSEARCH service is running and automatic population (for creating indices of the full-text catalog) is activated.

2.1.3 DB2

When using DB2 you have to create an operating system user. Afterwards a database user is created with the rights *connect to database* and *create tables*. Set the character set of the database to UTF-8 and the standard size of the buffer pool and table space to 16 KB. Then you create a database schema, for which the user is authorized.

2.1.4 PostgreSQL

Installation of PostgreSQL can vary a bit depending on the underlying platform.
For Windows, an installer is being used. During installation of PostgreSQL choose at least the following components:

- Database Server
  - Data Directory
  - National Language Support
- User interfaces
  - psql
  - pgAdmin III (optional admin GUI)
- Database Drivers : JDBC Driver

The windows-installer will display a "initialize database cluster" dialog, it is advisable to use UTF-8 as encoding and to check "accept connection on all interfaces" if remote connections to the database are needed.

In case of Linux as underlying platform, we recommend to use the package manager of your distribution to install the postgres package. Database creation and listener/interface specification will usually have to be performed manually afterwards; the steps will be described below as optionally.

Independent of the used platform (Windows/Linux), in the data subdirectory of the installation directory, edit the pg_hba.conf file to allow access from remote machines if desired, e.g.:

```
host  all  all  10.10.10.0/24  md5
```

In the data subdirectory of the installation directory edit the postgresql.conf file. Make sure that parameter default_with_oids is turned off:

```
default_with_oids = off
```

It is advisable to restrict the search path to the current user schema with:

```
search_path = '"$user"'
```

Optionally make sure that postgres listens on the appropriate network interfaces, if remote connections are needed:

```
listen_addresses = '*'
```

Then restart the postgres service:

- in Windows: via the service manager
- in Linux: via the appropriate command for the init framework of your distribution (e.g..: systemctl restart postgresql)

Then, as postgres user, the PgAdmin III gui or the psql command line tool can be used to

- Create a User Account ("Login Role" in Postgres Terminology)

  ```
  create role <EP_USER> login password '<eppasswd>'
  noinherit valid until 'infinity';
  ```

- Optionally create a database if no one was created at Postgres installation time, or if you want a separate one for **@enterprise**:

  ```
  create database <DBNAME> encoding='UTF8';
  ```

- Connect to the created database, either via the PgAdminIII gui or with psql:

  ```
  \c <dbname>
  ```

- Create a Schema: (preferably without any schema name, so the name of the schema will be the same as the name of the login role):

  ```
  create schema [<SCHEMANAME>] authorization <EP_USER>;
  ```

- Provide an appropriate default schema: If the names of the schema and of the login role must differ in your installation, be sure to
  - either set the default search path for the login role in the database via:

    ```
    alter role <EP_USER>
    in database <DBNAME>
    set search_path = <SCHEMANAME>;
    ```

  - or set the search path for sessions in the Session Environment field in install step 11 in section 2.2:

    ```
    set search_path=<SCHEMANAME>
    ```

To activate support for the soundex search, use the following command (the fuzzystrmatch*.sql files may be located in a separate package named postgres-contrib).

```
create extension fuzzystrmatch schema <SCHEMANAME>;
```

2.1.5 Derby

Derby doesn't need any preparation. Derby is perfectly suited for development purposes and test deployments. For heavyweight multiuser installations the use of Derby is not advisable.

2.2 Extract and Install

This section describes how to install the **@enterprise** stand-alone server. **@enterprise** is distributed on CD or can be downloaded from our web site. It is packed in one single file named setup100.jar. The installation can be started with a double-click on the file. The installation of a Java JRE 1.8 (or higher) is required.
If *.jar files are not associated with Java on your machine, or if you don't have a GUI available, please start the setup on a command line:

```
java -jar setup100.jar
```

The setup process consists of the following steps:

1. Verify if this is the version of **@enterprise** that you want to install and start the setup by clicking on *OK*.
2. Specify the directory of the Java compiler and interpreter.
3. Installation directory: The directory where the system will be installed.
4. Choose the port on which the **@enterprise** server will run.
5. If your server operating system is MS Windows you can install a service.
6. Now setup shows you information about how you can start the server and continue the setup process.
7. Setup will try to start the server and open a browser for you. If this fails and if you did not install a service, you have to start the server manually by executing the batch file (ep.sh or ep.bat).

   If your browser didn't already do it, please navigate to http://localhost:*port*/, where *port* is the port number that you have chosen during the previous setup steps. The rest of the installation is done with the browser.
8. The first screen is the Welcome-screen, click on *Start Setup* to start the configuration.
9. On the next screen you specify a logical name for the server (server ID), a server number (an integer value for distributed installations), the license key and the server's default language. Please note that changing the server id will result in invalid sessions after the first server restart (this applies to installations using the internal Jetty web-server; not to installations within an application server).
10. Now you can load a database JDBC driver. Use the *JDBC Driver Help Page* for information about different databases and their JDBC drivers.

11. On the next screen you have to specify some database parameters. We suggest to use the help function (the question mark next to *Database Type*) to fill the Database Type, JDBC Driver Class, and JDBC URL fields with valid values.

    - Database Type: The database; you can select ORACLE, DB2, MS SQL-Server, Firebird, or Derby.
    - JDBC Driver Class: Java class that contains the driver. Take a look at the table on page 38 for a list of driver classes.
    - JDBC URL: URL for the database. The syntax of this string depends on the JDBC driver used. See the examples on page 38 or consult the documentation of the driver. **@enterprise** allows to configure datasources too. For further information take a look in chapter 8.
    - Credentials not needed: Activate this checkbox, if database connection without credentials (user-id/password) should be used when authentication is accomplished externally (e.g. SQLServer Windows native authentication).
    - Database Userid: The ID of the user with whom you want to connect to the database.
    - Database Password: Password for the database user with the ID that you entered above.
    - Number of Connections: Default number of database connections.
    - Session Environment: You can specify SQL-commands, which are executed for each connection after connecting, for example: set TEXTSIZE 10000000 (SQLServer) or set search_path=ep (PostgreSQL)

    If SQLServer Windows native authentication is used, the following steps are needed:

    - If using sqljdbc driver, the file *sqljdbc_auth.dll* must be available in **@enterprise** root directory (or in case of service in <ep_root>/services directory).
    - Add to driver url the string integratedSecurity=true, e.g. jdbc:sqlserver://<host>:<port>;DatabaseName=<dbname>;integratedSecurity=true
    - Activate checkbox *Credentials not needed* and perform next setup step.

12. Now the database and driver will be tested. Optionally you can test if your database can store Unicode characters.

13. The next step is the creation of the database tables. The time may vary depending on your server's speed and the database that you use. If a schema of a (previous) **@enterprise** version exists, setup cannot be continued at this point!

14. After initializing the database, some internal services have to be started.

15. On the next screen the password of the system administrator can be specified.

16. Now a user and an organizational unit can be created. The following roles will be given to this user: *all*, *home* in the inserted organizational unit, and *sys*.

17. If you want, you can load an example process now.

18. Congratulation! You finished the setup of **@enterprise**. Click on *Login* to go to the login page, where you can immediately start to use **@enterprise**.

By completing the previous steps you finished the setup of **@enterprise**. If you want to change the configuration or configure advanced settings, take a look at chapter 3.

2.2.1 Bootstrap in stand-alone server (Jetty)

Since **@enterprise 8.0** the bootstrap mechanism is used, which builds the classpath automatically. This mechanism allows to keep the batch- and/or shell-file simple and clear. Following configurations are possible in classpath using *com.groiss.component.Bootstrap* in ep.bat or ep.sh:

The java property *-Dep.bootstrap.path* can be changed optionally, so additional paths can be added to classpath with following behavior:

- classes: all files within this folder are added to classpath
- lib: all files with extension **.jar* are added to classpath
- *.jar: the corresponding file is added to classpath
- all other paths are scanned for a *classes* and *lib* directory and the corresponding entries will be added to classpath. If these directories are not available, the entered directory will be added to the classpath.

Hint: The first entered path (leftmost) of property *-Dep.bootstrap.path* is loaded first, the rightmost path is loaded at last. The jar-files of the *lib* directory are loaded in alphabetical order.

Example:

```
%JAVACMD% -Xms16m -Xmx128m -Djava.awt.headless=true
  -Dep.bootstrap.path=C:/eproot;C:/extension/classes;../libs/lib;C:/myjar.jar;.
  com.groiss.component.Bootstrap conf\avw.conf
```

- C:/eproot is scanned for a *classes* and *lib* directory
- C:/extension/classes is added to the classpath
- ../libs/lib results in adding all included jar files to classpath (scanned relative to root-path)
- C:/myjar.jar is added to the classpath
- . means, that the root-path is scanned for a *classes* and *lib* directory. If these directories are not available, all elements of the root-path will be added to the classpath.

Hint: If property *-Dep.bootstrap.path* is set, only these paths/files will be considered, i.e. the default behavior of **@enterprise** classpath will be disabled.

2.3 Installing as a Windows Service

In Windows you can configure a stand-alone installation of **@enterprise** to run as service. This can be done while installing (see the previous section) or later by calling the service install script in the service subdirectory of **@enterprise**
@enterprise uses the *procrun* framework which is part of the Apache Commons Daemon Project.

2.3.1 Components of the Framework

The service framework consists of the following files in the service subdirectory:

Common service framework files

- eprunsrv32.exe, eprunsrv64.exe: The core process runner wrappers, renamed from the Apache original distribution.
- eprunmgr.exe: GUI application for monitoring and configuring *procrun* services (also renamed from the Apache project).
- Elevate32.exe, Elevate64.exe: Admin rights utility for Installations with User Account Control (UAC); c.f. sudo in Linux.
- service.bat: Used to install, update and delete the service. Can be called in any of the following modes:
  - service install: Installs the service. This is the only variant that is being called (during initial installation). All other variants are for manual usage via the command line.
  - service delete: Deletes the service.
  - service update: Updates parameters of the service.
  - service edit: Starts a GUI to edit parameters of an existing service.
  - service monitor: Present a GUI to monitor the service.

  All those calls must be performed with the proper permissions when UAC is enabled, i.e. prefix the calls with ElevateXX. Updating and deleting a service should only be attempted when the service is not running currently. It is also recommended to abstain from using/opening either the service manager or registry editor when installing, updating and deleting.

  Details for service parameters to change within service.bat can be found at https://commons.apache.org/proper/commons-daemon/procrun.html.

Utility to send a CTRL-BREAK signal to a process

SendSignal.exe

Use

[Elevate32] SendSignal <pid>

to get a threaddump. The <pid> is the PID of the wrapper process. The thread-dump is written to the commons daemon log file (as specified via the *–LogPath* and *–LogPrefix* parameters in service.bat. ElevateXX is needed in case of UAC.

2.3.2 Migrating to the new procrun framework

Before June 2017, the service framework for **@enterprise** has been the Tanuki Java Service Wrapper. While this was an adequate, well working framework, there were technical issues with it when running under 64-bit Windows installations, as well as license issues, since the free community editions are restricted to 32-bit environments.
The Tanuki framework was configured via entries in the wrapper.conf file. The procrun wrapper is configured via the command line. The essence is captured in the service.bat file:

Java directory and service name

The most important parameters in the service.bat script are the lines

```
set "JAVADIR=%javadir_placeholder%"
set "SERVICEID=%servicename_placeholder%"
```

which should have been replaced at installation time with the path to the java installation directory to be used and with the service name, e.g.:

```
set "JAVADIR=C:\Program Files\Java\jre1.8.0_131"
set "SERVICEID=@enterprise90"
```

The javadir_placeholder can be taken from the wrapper.java.command in wrapper.conf: use the value up to, but excluding \jre or \bin as substitution for %javadir_placeholder%. The servicename_placeholder corresponds to the value of the wrapper.ntservice.name property of wrapper.conf.

Memory parameters

Memory limits are specified via the –JvmMs and –JvmMx lines in service.bat. The values should be taken from the wrapper.java.initmemory and wrapper.java.maxmemory properties in wrapper.conf.

Additional parameters

Additional arguments for the Java VM, which were in wrapper.java.additional.* properties of wrapper.conf should be placed in separate ++JvmOptions lines in service.bat.

ServiceDependencies

Dependencies on other services, which where stated in wrapper.ntservice.dependency.* properties of wrapper.conf are to be placed in separate ++DependsOn lines in service.bat.

2.3.3 Migration steps

Execute the following sequence of steps:

- Adapt the service.bat: According to the ruled given above.
- Stop the old service: Use the service manager or sc to stop the old **@enterprise** service:

  sc stop <ServiceName>

- Uninstall the old service:

  sc delete <ServiceName>

- Install the new service:

  service install

- Start the new service:

  sc start <ServiceName>

  Make sure the new service is running properly, check the logfile log\commons-daemon-*.log and the parameters.

- Remove obsolete files: The install.bat, uninstall.bat and run.bat files as well as the wrapper.* files can be removed from the service subdirectory.

2.3.4 Registry entries

In the registry, the service installation places the main properties at
HKLM\SYSTEM\CurrentControlSet\Services\<ServiceName>
Additional properties can be found at as well as:
HKLM\SOFTWARE\Apache Software Foundation\ProcRun 2.0\<ServiceName>\Parameters
or on 64-bit Machines
HKLM\SOFTWARE\Wow6432Node\Apache Software Foundation\ProcRun 2.0\<Service-Name>

2.4 Installing as a Linux Daemon

For systemd based Linux distributions, there is a quite simple pattern to use **@enterprise** as a daemon without any additional overhead. We provide a template unit file enterprise.service located in the service subdirectory of the **@enterprise** installation.
Copy this file to your local systemd directory (usually at /etc/systemd/system).
Modify it according to your needs:

- replace %epuser% with the user id of the Linux user[1] that should run **@enterprise**.
- replace %epworkdir% with the absolute canonical path[2] to the **@enterprise** installation directory.
- replace %epjava% with the absolute canonical path to the java executable.
- state appropriate dependencies on other services for startup (e.g. dependency on DBMS) in the Requires line.
- adapt the arguments of the command line (be sure to use ␣\) at the end of each line).

Your enterprise.service unit description file could look like this:

```
#[Unit]
Description=@enterprise BPMS
After=syslog.target network.target
#Requires=a.service b.service

[Service]
User=epadm
Group=users
Type=simple
Restart=on-failure
RestartForceExitStatus=2
WorkingDirectory=/opt/ep/ep90
ExecStart=/usr/java/jdk1.8.0_131/jre/bin/java \
-classpath lib/bootstrap.jar \
-Xms32m -Xmx256m  \
-Djava.awt.headless=true \
com.groiss.component.Bootstrap \
conf/avw.conf

[Install]
WantedBy=multi-user.target
```

After changing the file, its recommended to reload the systemd daemon by means of

```
systemctl daemon-reload
```

The service can then be administrated with the usual systemd stanzas:

- enable startup of **@enterprise** at system startup:

  ```
  systemctl enable enterprise
  ```

- disable startup of **@enterprise** at system startup:

[1] please note that privileged ports (port numbers below 1024) can only be opened when run with User=root.
[2] such a path starts at / and does not follow any symbolic links

```
systemctl start enterprise
```

- enquire the state of **@enterprise**:

```
systemctl status enterprise
```

- start **@enterprise** manually:

```
systemctl start enterprise
```

- stop **@enterprise** manually:

```
systemctl stop enterprise
```

- restart **@enterprise** manually:

```
systemctl restart enterprise
```

2.5 Using an Application Server or Servlet Container

If you want to run **@enterprise** in an application server (e.g., IBM's *WebSphere*) or a servlet container (e.g., Apache's *Tomcat*) you need the **@enterprise** web application archive file named ep100.war which is especially prepared for this purpose. Deploy this file in your server. Afterwards, open your browser and navigate to http://host:port/context-root/, where host and port must be the right values for accessing your server and context-root is the context root that you chose when deploying the file. See section 2.2 step 8 for details about the rest of the installation.

Hint: The database name of the Derby JDBC-URL in embedded mode (the 'ep' part in *jdbc:derby:ep;create=true*) is relative to either the current directory or relative to the directory specified in the *derby.system.home* system-property (if this is present). In a scenario of deploying multiple **@enterprise** systems as different web-applications in one servlet-container, with each of the systems using a dedicated embedded derby instance, use unique path names to the database files per web-application
(e.g.
jdbc:derby:databases/app1/derbydb;create=true and
jdbc:derby:databases/app2/derbydb;create=true).

Hint: Uploading of driver jar files during installation might not work satisfactorily depending on classloading implementation details and preinstalled driver jar files in some common area of the application server. If you want a specific version of a driver, it is advisable to change the underlying driver jar in the application server. If this is not feasible, try to include the desired driver jar file in the ep*.war file.
This might not be sufficient for your particular application server. E.g. since Weblogic comes with its own version of Derby, for the combination of Weblogic as application server and Derby as database management system it is also required to specify the class loading order by adding the org.apache.derby.* packages name to the prefer-application-packages element located in WEB-INF/weblogic.xml file in the ep*.war.

Hint: If Tomcat is used as Servlet Container and UTF-8 encoding should be used, you have to set following attributes in Tomcat's *server.xml*:

- URIEncoding="UTF-8"
- useBodyEncodingForURI="true"

Typically:

```
<Connector port="8080"
 maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
 enableLookups="false" redirectPort="8443" acceptCount="100"
 debug="0" connectionTimeout="20000"
 URIEncoding="UTF-8" useBodyEncodingForURI="true"
 disableUploadTimeout="true" />
```

Hint: The Servlet API 3.1 is used! This can lead to compatibility problems with some application server, e.g. Tomcat in versions less than 8.0.

2.6 Unattended installation

@enterprise offers the possibility to create a complete installation without additional user interaction. This could be necessary, if many **@enterprise** systems must be deployed with same installation files, but with different configuration settings.
Example: A test system, a staging system and a production system are needed. All systems have the same applications to deploy, but with different database and mail-server settings.

As point of origin the setup files (setup*.jar for standalone or ep*.war for application server) provided by Groiss Informatics GmbH should be taken and enriched with own files for configuring/installing **@enterprise** itself and its applications. The following sections describe the preparation and deployment of such an installation.

2.6.1 Preparation of installation files

As point of origin the following artifacts are needed:

1. Depending on the usage of **@enterprise** as standalone server (Jetty) or in an application server (see section 2.5) the appropriate setup file is needed.
2. The database driver file.
3. The local configuration of **@enterprise** and as needed the configuration of the applications (see section 2.6.2 for details).
4. The installable ZIP files of applications (for details how to package applications take a look into the **@enterprise** Application Development Guide - sections *5.1 Organization of Files*, *5.2 The Configuration File* and *5.7 Installation* or check out the *demos.zip* delivered with **@enterprise**).

5. Other needed files for installation, e.g. application-independent xml-imports or csv-files (application-dependent files should be part of application zip files).

6. A groovy-script file which performs the installation of the applications (see section 2.6.3 for details).

The files of items 2-6 must be added to the appropriate setup file with a ZIP tool or the jar-command of the JAVA distribution as shown in following examples:

```
jar -uf setup100.jar conf/avw.conf lib/db_driver.jar appls
jar -uf ep100.war WEB-INF
```

File structure for standalone installation (Jetty)

```
appls
  appl1.zip
  appl2.zip
  ...
  dept_import.xml
  installappls.scr
conf
  avw.conf
lib
  db_driver.lib

setup.jar
```

File structure for installations in application server

```
WEB-INF
  appls
    appl1.zip
    appl2.zip
    ...
    dept_import.xml
    installappls.scr
  conf
    avwservlet.conf
  lib
    db_driver.lib

ep.war
```

2.6.2 Define configuration

For the configuration a file has to be created (avw.conf or avwservlet.conf) which contains the parameters for **@enterprise** and the applications. This file is taken as configuration file (avw.conf or avwservlet.conf) for **@enterprise**. The application specific parameters are

extracted and added (or overwritten) to the appropriate "appl.prop" files.

Mandatory parameters for **@enterprise** are

- avw.license: The license key (see section 3.1).
- avw.servername: The name of the server.
- Database parameters: At least the set of parameters as defined in the example below (see section 3.3 for details).
- services: Contains only the install service *com.groiss.server.InstallService inst* for unattended installation
- services.standard: The services for running a successful installed system. The values of this parameter are copied to parameter *services* and *services.standard* is removed after a successful installation (see section 3.6 for details).
- httpd.port for standalone installations: The port on which the standalone server runs (see section 3.2 for details). This parameter is not needed for installations in application server.

Optionally with parameter database.drop.all=true you can define, if an existing database schema should be dropped during install process.

The parameters of the applications must contain the following prefix (a backslash before the colon is needed!): <appl_id>\:

Example for a configuration file:

```
# required parameters

avw.license=<license_key>
avw.servername=ep10

database=com.dec.gi.sql.DBOracleLOB
database.driver.class=oracle.jdbc.OracleDriver
database.url=jdbc\:oracle\:thin\:@localhost\:1521\:mydb
database.user=<ep_user>
database.password=<ep_pwd>
database.connections.max=8

services.standard=com.groiss.store.DBConnPool store,
                  com.groiss.server.ApplicationLoader appls,
                  com.groiss.server.EventDispatchServer ev,
                  com.groiss.httpd.jetty.Jetty httpd,
                  com.groiss.reporting.ReportingService reporting,
                  com.groiss.timer.TimerManager timer
services=com.groiss.server.InstallService inst
```

```
# optional parameters

logger.trace=DEBUG
httpd.maxthreads=25
httpd.minthreads=2
httpd.port=8090

Locale.list=en_US,en_GB,de_AT
Locale.language=en
Locale.country=GB
ep.product.name=WFM system
passwdpolicy.days_password_valid=180
passwdpolicy.days_warning_before=10
passwdpolicy.max_count_invalid_logins=5
passwdpolicy.min_length=8
passwdpolicy.min_capitals=1
passwdpolicy.min_digits=2
passwdpolicy.min_others=1
passwdpolicy.history_steps=10


# appl1 parameters
appl1\:param1=val1
appl1\:param2=val2


# appl2 parameters
appl2\:param1=val1
appl2\:param2=val2
```

2.6.3 Define install script

The deployment of applications or all other operations in **@enterprise** which cannot be handled within a configuration (e.g. import csv-files, etc.) can be achieved with a groovy-script file within the appls-directory. The name of the file must be *installappls.scr*! The groovy context is the same as for the administration shell component (see Administration Guide - section *12 Administration Shell*).

Example for a script file:

```
//install the application "appl1"
f = new File(com.groiss.util.Settings.getBaseDir(), "/appls/appl1.zip");
admin.installApplication("appls/appl1", f);

//install the application "appl2"
f = new File(com.groiss.util.Settings.getBaseDir(), "/appls/appl2.zip");
```

```
admin.installApplication("appls/appl2", f);

//import application independent file
sw = new StringWriter();
is = new FileInputStream(new File(com.groiss.util.Settings.getBaseDir(),
                                  "/appls/dept_import.xml"));
new com.groiss.impex.Import().importXML(is, new PrintWriter(sw));
```

2.6.4 Perform installation

The installation of a complete packaged setup file can be achieved in following ways:

- Standalone: Perform the JAVA call in command line

  java -Djava.awt.headless=true -jar setup.jar <dest_dir> <java_dir>

  The <dest_dir> is the destination directory where **@enterprise** should be extracted. The <java_dir> defines the location of Java installation directory; if the keyword "default" is entered, the default location of JAVA is taken.

- Application server: Deploy the WAR-file in application server.

For both ways in first step the content of the setup files is extracted. In case of standalone the server is started immediately after extracting with the install service (*com.groiss.server.InstallService*). In case of application server the **@enterprise** servlet *AVWInit* is loaded before the install service starts. The install service loads the configuration and extracts the application parameters into individual files (<appl_id>_appl.prop) within the appls-directory. After this operation the database is initialized and then the install script (installappls.scr) is executed. As last step the application parameters are set/merged, the **@enterprise** configuration parameter setup is set to true and the standard services are started. After successful installation process the log must contain the message "InstallService completed successfully." - we recommended to restart the **@enterprise** server again!

2.7 Basic considerations for backup and recovery

According the operational aspects of backup and recovery, an **@enterprise** installation comprises of the following component types:

- Basic **@enterprise** software artefacts
- Application specific software artefacts
- Configuration data
- Application Logfiles
- Database content

For the first two types, quite ordinary backup and recovery measures are perfectly appropriate. The small volume and infrequent changes to this components allow for periodic full backups of the **@enterprise** installation directory and the application installation directory. Configuration data comprises a small set of very small configuration files (avw.conf in the conf subdirectory of the **@enterprise** installation directory and appl.prop in the application directories). Changes to those files will be relatively infrequent (after an initial production phase), but might be critical to proper operation. Frequent or even immediate backup of those files is recommended, possibly by incorporating them in a version control system.
Application logfiles (usually in the log subdirectory of the **@enterprise** installation directory) should be rotated and a periodic or rotation triggered copy could be made. The logfiles are not essential for the operation, so there is no need to recover them, nevertheless they might be essential for gaining insight of the nature of the problem and help to avoid further errors.
Concerning the database, backup and recovery measures are obviously vendor specific. But some general remarks are nevertheless applicable. Periodical backup of the data files is strongly recommended. To be able to recover to the youngest point in time possible, transaction log writing must be enabled. The logs must be switched and shipped to a safe location on the fly.
Concrete recovery measures depend on the type and range of the disaster, but in general, they consist of recovering the database from the latest backup of data files and transaction logs, to copy the software artefacts of **@enterprise** and of the application back to their destinations and to reinstitute the configuration data.

3 Configuration

This chapter describes advanced configuration parameters of **@enterprise**. You can change the data that you entered at setup as well as additional configuration here. Open the configuration area in the system administration by clicking on *Configuration* in the menu on the left side.

In order to save your changes, you must use the *Save* icon in toolbar, which is available on every configuration page. After activating this button, the changes are stored in the file *avw.conf* by default, which can be found in the folder *classes* of **@enterprise** installation directory. **@enterprise** offers the possibility to define another configuration files. For this purpose the batch-/shell-file or wrapper.conf for starting **@enterprise** has to be extended by a comma-separated list (or alternative by file path separator ; in Windows and : in Linux) of filenames or whole folders, e.g. add after *com.groiss.component.Bootstrap* "conf/avw.conf,conf/myavw.conf,conf2,my conf". In this example *avw.conf* and *myavw.conf* are files, *conf2* and *my conf* are folders and contain a various number of configuration files. All configuration files of a folder must contain the prefix *.conf which are read in alphabetical order (case-insensitive). If a path to configuration files/folders contains spaces, the whole string must start/end with apostrophe as shown in example above. If a parameter is available in more the one configuration file, the first appearance will be taken. When changing settings via GUI, no server restart is necessary, excepting the notification icon appears (yellow triangle)!

Each parameter has a value which is set by default. If the entered value is different to the default value, an icon appears for resetting the value. After activating this icon the *Save* icon must be clicked to persist the changes.

In the following we describe the different parameter groups. Each of them is represented by an entry in the configuration menu. If you use a German server installation and encounter problems understanding the English terms used in this manual, we suggest to create and use an administrator with English language (the *sys* right is required in order to enter the administration).

Hint: The parameter definition and their groups are defined in *properties.xml*. This file should not be changed!

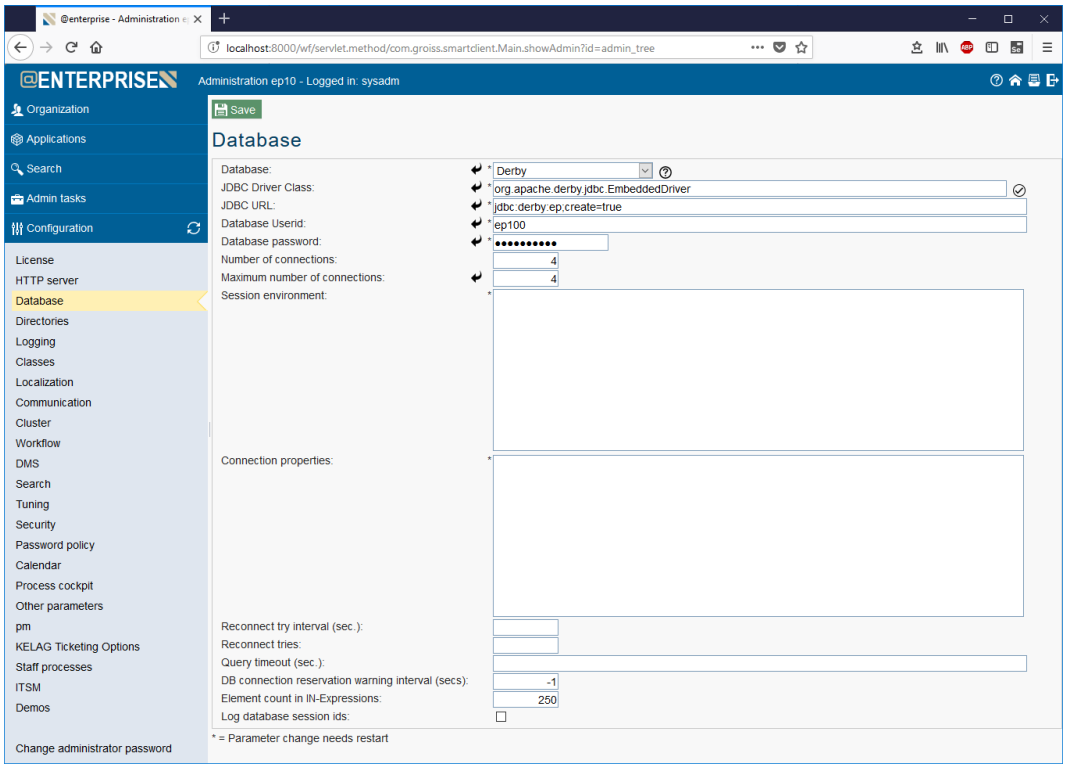

Figure 3.1: **@enterprise Configuration**

3.1 License

The first screen contains license information:

- **License key - avw.license:** Your license key. If you want to change your license key after you finished the setup, you can enter the new key here.

3.2 HTTP server

This screen contains the setup of the HTTP server:

- **Server IP port - httpd.port:** HTTP port on which the server runs.
- **IP address - http.ip-address:** The default-behavior of multiple network-interfaces: the HTTP-server runs on all interfaces. With this parameter you can restrict the interfaces by entering an ip-adress, where the server should run.
- **Minimum number of threads - httpd.minthread:** Number of threads, which are started on startup.
- **Maximum number of threads - httpd.maxthreads:** Maximum number of threads, which will be used for HTTP requests.

  Hint: If Apple Safari is used in combination with SSL, it is recommended to set an adequate high number for *Minimum Number of Threads* and *Maximum Number of Threads*.
- **Server SSL Port - ssl.port:** Port of the HTTPS server.
- **SSL IP address - ssl.ip-address:** Analog to parameter *IP address*, but for SSL port.
- **Client certificates for HTTPS - sl_requireclientcertificate:** This parameter determines how a secure SSL connection can be established by a client. There are three possibilities:
  - **Are not requested:** If this option is selected, SSL connections are established in any case.
  - **Are required:** If this option is selected, SSL connections are established only if the client has a valid certificate for authorization.
  - **Are requested:** If this option is selected, the establishment of SSL connections depends on the content of the response: if the response contains a valid client certificate the SSL connection is established automatically; if the response contains no valid client certificate a login mask will be displayed to the user and after a successful login the SSL connection will be established.
- **Administrative IP port - httpd.admin.port:** This port is used for administrating **@enterprise** - a admin session will be created which is necessary to execute administration functions. If no port is defined, the current user is already logged in and if he want to change to administration, he will be requested to log-in again (for getting admin session).

- **Administrative IP address - httpd.admin.ip-address:** Define an own IP address for administration. The behaviour is analog to parameter *Administrative IP port*.
- **Use SSL for administration - httpd.admin.usessl:** If activated, the *Administrative IP port* is a SSL port. If no port is defined, the *Server SSL Port* is used.
- **Allowed hosts or networks for administration - ep.adminshell.allowedips:** A list of hosts and networks can be specified. These hosts can access the administration of HTTP server. The syntax of this field is described below in section 3.2.1.
- **Allowed hosts or networks - httpd.hosts.allow:** Analogous to *Allowed hosts or networks for administration*, but only for non administration session. If *Allowed hosts or networks for administration* is empty and this host/network matches, it is possible to enter the administration (session).
- **Denied hosts or networks - httpd.hosts.deny:** Analogous to above, but only for non administration session.
- **Access control - urls.allowed:** We provide a mechanism which allows to grant or deny access to method-URLs based on a combination of IP-addresses and rights. The syntax of access rules and their semantics is described below in section 3.2.2.
- **Exclude SSL ciphersuites - httpd.jetty.sslconnector.excludeciphersuites:** Vulnerable SSL cipher suites can be excluded from use in HTTPS with following line:

```
httpd.jetty.sslconnector.excludeciphersuites=
TLS_RSA_WITH_AES_128_CBC_SHA,
\r\nTLS_RSA_WITH_AES_256_CBC_SHA,
\r\nTLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
\r\nTLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
\r\nTLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
\r\nTLS_ECDH_RSA_WITH_AES_256_CBC_SHA,
\r\nTLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
\r\nTLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
\r\nTLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
\r\nTLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
\r\nTLS_DHE_RSA_WITH_AES_128_CBC_SHA,
\r\nTLS_DHE_RSA_WITH_AES_256_CBC_SHA,
\r\nTLS_DHE_DSS_WITH_AES_128_CBC_SHA,
\r\nTLS_DHE_DSS_WITH_AES_256_CBC_SHA,
\r\nSSL_RSA_WITH_3DES_EDE_CBC_SHA,
\r\nTLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
\r\nTLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
\r\nTLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
\r\nTLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
\r\nSSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
\r\nSSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
\r\nSSL_RSA_WITH_DES_CBC_SHA,
\r\nSSL_DHE_RSA_WITH_DES_CBC_SHA,
```

\r\nSSL_DHE_DSS_WITH_DES_CBC_SHA,
\r\nSSL_RSA_EXPORT_WITH_RC4_40_MD5,
\r\nSSL_RSA_EXPORT_WITH_DES40_CBC_SHA,
\r\nSSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA,
\r\nSSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA,
\r\nSSL_RSA_WITH_NULL_MD5,
\r\nSSL_RSA_WITH_NULL_SHA,
\r\nTLS_ECDH_ECDSA_WITH_NULL_SHA,
\r\nTLS_ECDH_RSA_WITH_NULL_SHA,
\r\nTLS_ECDHE_ECDSA_WITH_NULL_SHA,
\r\nTLS_ECDHE_RSA_WITH_NULL_SHA,
\r\nSSL_DH_anon_WITH_RC4_128_MD5,
\r\nTLS_DH_anon_WITH_AES_128_CBC_SHA,
\r\nTLS_DH_anon_WITH_AES_256_CBC_SHA,
\r\nSSL_DH_anon_WITH_3DES_EDE_CBC_SHA,
\r\nSSL_DH_anon_WITH_DES_CBC_SHA,
\r\nTLS_ECDH_anon_WITH_RC4_128_SHA,
\r\nTLS_ECDH_anon_WITH_AES_128_CBC_SHA,
\r\nTLS_ECDH_anon_WITH_AES_256_CBC_SHA,
\r\nTLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,
\r\nSSL_DH_anon_EXPORT_WITH_RC4_40_MD5,
\r\nSSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA,
\r\nTLS_ECDH_anon_WITH_NULL_SHA,
\r\nTLS_KRB5_WITH_3DES_EDE_CBC_SHA,
\r\nTLS_KRB5_WITH_3DES_EDE_CBC_MD5,
\r\nTLS_KRB5_WITH_DES_CBC_SHA,
\r\nTLS_KRB5_WITH_DES_CBC_MD5,
\r\nTLS_KRB5_EXPORT_WITH_RC4_40_SHA,
\r\nTLS_KRB5_EXPORT_WITH_RC4_40_MD5,
\r\nTLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA,
\r\nTLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5

The cipher suite used by a client can be seen via the URL
...servlet.method/com.dec.avw.html.HTMLAdmin.clientInfo.

Dealing with sporadic SSL-Handshake problems is greatly eased by setting the javax.net.debug system property in the java command line, eg.:

-Djavax.net.debug=ssl:defaultctx:sslctx:handshake:verbose

This generates considerable amounts of log data, usage is only advisable when client connection issues via HTTS arise. An on-line assessment of your SSL parameters can be obtained at https://www.ssllabs.com/ssldb/index.html

- **Exclude SSL Protocols - httpd.jetty.sslconnector.excludeprotocols:** Protocols for Jetty SSL connectors can be specifically excluded. Please note that the default protocols being used depend on the specific Java version and release being used.

- **Context path - avw.contextpath:** This parameter defines the context path of **@enterprise**.

3.2.1 Defining Allowed and Denied Hosts or Networks

To restrict access to the HTTP server to selected hosts or address ranges you can declare an *allow* and a *deny* list. The evaluation is as follows: If the allow-list is empty, access is allowed from every host except the ones in the deny-list. If the allow-list is not empty, access is allowed from the hosts and networks in the allow list minus the hosts (and networks) in the deny list.
Both lists contain pairs of IP-Addresses and net-mask separated by spaces, commas or newlines. Both IPV4 and IPV6 addresses are permissible. A net-mask should be given in the CIDR style in form of an integer specifying the number of bits of the network-part. For IPV4 addresses, the traditional dotted notation is also permitted. See the following example:

```
10.205.112.0/255.255.255.0
P10.205.113.0/255.255.255.0
10.205.224.0/24
2001:0db8:0010::/48
```

This entries in the allow-list means, access from the networks 10.205.112.*, (proxy-address) *10.205.113.0*,10.205.224.* and 2001:0db8:0010:* is allowed. When entering IPV6 addresses directly in the config-file, bear in mind that each colon (:) must be escaped by preceding it with a backslash.
The following list used for the allow-list causes that access from hosts 10.205.112.4, 10.205.224.8 and 2001:DB8:0010:0:8:800:200C:417A is allowed.

```
10.205.112.4/32
10.205.224.8/255.255.255.255
2001:DB8:0010:0:8:800:200C:417A/128
```

3.2.2 Access Control

The access control mechanism affects only the Dispatcher which serves URLs targeting java methods. Rules can be specified which restrict access to certain URLs based on a combination of IP-address and **@enterprise** rights.
To activate the access control, the corresponding service must be added to the services in *Classes* → *Services* :

```
com.groiss.avw.contrib.URLChecker uc
```

Configuration

The access control property consists of a comma-separated list of rules. Each rule combines an IP-specifier, an URL-prefix and a set of rights separated by spaces. Each of the components can be a wildcard in the form of an asterisk.
Accordingly, the syntax of the ruleset is:

{ (["P"] ip-specifier | "*") SPACE (url-prefix | "*") SPACE ("*" | "DENY" |
(right { SPACE right }*) COMMA }*

The optional "P" before the ip-specifier designates a proxy-address.
Without specifying the "P", the remote address is the leftmost entry in "X-Forwarded-For" header field (if it exists), and the HttpRequest.getRemoteAddress(), if the header field does not exist.
When "P" is specified, the match is performed just against the HttpRequest.getRemoteAddress() without taking into account any "X-Forwarded-For" header.
The IP-specifier consists of an ip-address and a net-mask separated by a "/". Both IPV4 and IPV6 addresses are permissible. A net-mask should be given in the CIDR style in form of an integer specifying the number of bits of the network-part. For IPV4 addresses, the traditional dotted notation is also permitted. It can be used to specify a single host or a subnet in the following ways

10.205.112.22/255.255.255.255	designates the single host 10.205.112.22
10.205.224.22/32	designates the single host 10.205.224.22
P10.205.112.23/255.255.255.255	designates the proxy host 10.205.112.23
10.205.112.0/255.255.255.0	designates all hosts in the subnet 10.205.112.*
10.205.224.0/24	designates all hosts in the subnet 10.205.224.*
10.0.0.0/255.0.0.0	designates all hosts in subnet 10.*.*.*
11.0.0.0/8	designates all hosts in subnet 11.*.*.*
2001:0db8:0010::/48	designates all hosts in subnet 2001:0db8:0010:*
::ffff:0a0a:0a0a/128	designates a single IPV4 hosts 10.10.10.10
*	this wildcard designates all hosts

Technically, the IP-address of a requester matches an IP-specifier when the network prefix denoted by the netmask matches.
An URL-prefix consists of the first characters of a fully qualified method name (package, class, method). The URL-prefixes are case sensitive.

com.groiss	designates all calls to methods in classes in packages located in com.groiss or below
com.groiss.org.PasswdAuth	designates all calls to methods in the class com.groiss.org.PasswdAuth
*	this wildcard designates all methods regardless of origin

The set of rights is a space separated list of IDs of **@enterprise** rights. The right IDs are case sensitive.

set_agent	designates all users who have the right set_agent
admin stat	designates all users who have the right admin and / or the right stat
*	wildcard designating that rights are not needed
DENY	special dummy right id, can be used to deny access

Examples for Rules

The following examples show how those three designations can be combined to form a rule:

127.0.0.0/8 * *
Access from local host subnet is not restricted.

10.205.112.26/32 * DENY
Access from 10.205.112.26 is not allowed.

10.205.112.0/24 com.groiss.org.PasswdAuth *
Login of hosts from subnet 10.205.112.0 is allowed.

10.205.112.0/24 * internal
All operations of hosts from this subnet are allowed if users have the right internal.

* com.groiss DENY
Access to com.groiss.** classes and methods is denied to every host.

* com.my.appl admin,customer
Access to com.my.appl.* classes and methods is allowed if users have the right admin or customer.

* * DENY
Deny everything from everywhere.

Semantics

The validation of a list of rules in the *Access Control* property is as follows:
If the property is empty, nothing is filtered.
Otherwise all rules are checked in the order they are defined until a rule matches according to IP-specifier and URL-prefix. For a matching rule, the validation depends on the set of rights of the rule. We distinguish two cases:

- **Existing Session** (user already logged in):
  The intersection of the rights of the user and the rights given in the rule is computed. If the intersection is empty, access is denied (an exception is thrown), else the rule succeeds and access is granted.
- **No Session** (user not yet logged in):
  If the set of rights of the rule consists of a single DENY element, then access is denied (an exception is thrown), else the rule succeeds and access is granted.

If no rule at all matched, access is granted. This can be avoided if the last rule is "* * DENY".

Other Operational Considerations

Access Control gets reconfigured if the configuration is changed. This is also logged at log level 1 to allow one to find incorrect rules. Normal operations of *Access Control* are logged at log level 3.
Access Control is not automatically aware of additional rights given to a user or role or to the revocation of rights from them. In order to know about the constellation, the affected users must log out and log in again or the configuration must be saved (thereby reconfiguring *Access Control*). Caching of user rights in the *Access Control* mechanism is logged at log level 2.

3.3 Database

We suggest to use the help function (the question mark next to *Database*) to fill the Database, JDBC Driver Class, and JDBC URL fields with valid values for a selectable database.

- **Database - database:** The database; you can select ORACLE, DB2, MS SQL-Server, Firebird, or Derby.
- **JDBC Driver Class - database.driver.class:** Java-Class, that contains the driver. See the table on page 38 for a list of driver classes.
- **JDBC URL - database.url:** URL for the database. The syntax of this string depends on the JDBC driver used. See the examples on page 38 or consult the documentation of the driver.
- **Database Userid - database.user:** The ID of the user with whom you want to connect to the database.
- **Database password - database.password:** Password for the database user with the ID that you entered above.
- **Number of connections - database.connections:** Default number of database connections.
- **Maximum number of connections - database.connections.max:** The maximum number of database connections that can be created.
- **Session environment - database.session.env:** You can specify SQL-commands, which are executed for each connection after connecting, for example: `set TEXTSIZE 1000000`
- **Connection properties - database.connection.properties:** You can specify e.g. SSL properties to establish a secure connection to database. The value of this property is a list of property declarations separated by \r\n. Note that the = sign must be escaped by \ when editing directly in *avw.conf*.

  e.g. database.connection.properties=my.prop\=a.value\r\nyour.prop\=another.value

- **Reconnect try interval (sec.) - database.waitFor.seconds:** Interval in seconds for reconnect tries to the database.
- **Reconnect tries - database.waitFor.count:** Number of reconnect tries.
- **Query timeout (sec.) - database.query.timeout:** Number of seconds after which a query times out.
- **DB connection reservation warning interval (secs) - database.connection.busy.warning.secs:** Long-lasting reservations of DB connections can be logged and also monitored via the Server Monitor (Aged DB connections). Information includes thread-name, timestamp and stacktrace at moment of reservation. Monitoring information in the logfile will occur in 2 minute intervals. Each long-lasting reservation is logged not more than once. Following values can be defined:
  - -1 : do not monitor (default, behavior like before)
  - >=0 : do monitor; log /report all connections being reserved longer than the specified time interval (seconds)
- **Element count in IN-Expressions - ep.inlists.splitsize:** **@enterprise** uses SQL "IN-lists" - SQL expressions of the form WHERE att IN (val1, val2, ..., valn) as one form of optimizing database access.

  The API provides the com.groiss.store.BulkQuery class as a convenient means to utilize this fast kind of access.

  Since database systems usually put restrictions on the textual length of SQL-statements and also on the number of elements in such IN-lists, **@enterprise** splits queries with long IN-expressions into several queries. This configuration parameter can be used to control the maximum number of elements of an IN-expression.

  The default value is 250 elements. Increasing the parameter leads to fewer partial queries and fewer roundtrips to the database, but also to longer statements and IN-lists with the possibility to hit the limits imposed by your DBMS.
- **Log database session ids - database.logdbsessionid:** Check this parameter to log session ids of database (Oracle, SQLServer). To include database session ids in the log, it is necessary, that the database user *SYS* executes the following grant:

  ```
  grant select on v_$session to ep;
  ```

Table 3.1 shows the recommended drivers for the databases, their class names and JDBC URLs (you can directly view and use this table in **@enterprise** by clicking on the help link next to *Database*).

3.4 Directories

Here you can define some directories that **@enterprise** will use. The *Directory of Form Classes* and *Directory for Temporary Files* must exist.

DBMS	Driver Vendor	Driver Kind	Class and URL
DB2 UDB	IBM	Data Server Driver	com.ibm.db2.jcc.DB2Driver jdbc:db2://'host':50000/'dbname'
DB2 Z/OS	IBM	OS390	COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver jdbc:db2os390:'location-name'
Derby	Apache	Embedded	org.apache.derby.jdbc.EmbeddedDriver jdbc:derby:ep;create=true
Firebird SQL 1.5	Firebird	JCA	org.firebirdsql.jdbc.FBDriver jdbc:firebirdsql:'host'/3050:'dbalias'
MS-SQLServer (V2005+)	Inetsoftware	Una2000	com.inet.tds.TdsDriver jdbc:inetdae:'host':1433?sql7=true
MS-SQLServer (V2005+)	jTDS Project	jTDS	net.sourceforge.jtds.jdbc.Driver jdbc:jtds:sqlserver://'host':1433/'dbname'
MS-SQLServer (V2005+)	Microsoft	V3.0+	com.microsoft.sqlserver.jdbc.SQLServerDriver jdbc:sqlserver://'host':1433;database='dbname'
MySQL (V5.0, experimental)	MySQL	Connector/J (3.1)	com.mysql.jdbc.Driver jdbc:mysql://'host':'port'/'dbname'
Oracle LOBs	Oracle	Thin (V10g+)	oracle.jdbc.OracleDriver jdbc:oracle:thin:@'host':1521:'SID'
Oracle LOBs	Oracle	OCI	oracle.jdbc.OracleDriver jdbc:oracle:oci:@'TNSNAME'
PostgreSQL (V8.1+)	PostgreSQL	Native	org.postgresql.Driver jdbc:postgresql://'host':'port'/'database'
MS-SQLServer (-V2000)	Inetsoftware	Una2000	com.inet.tds.TdsDriver jdbc:inetdae:'host':1433?sql7=true
MS-SQLServer (-V2000)	jTDS Project	jTDS	net.sourceforge.jtds.jdbc.Driver jdbc:jtds:sqlserver://'host':1433/'dbname'
MS-SQLServer (-V2000)	Microsoft	V3.0+	com.microsoft.sqlserver.jdbc.SQLServerDriver jdbc:sqlserver://'host':1433;database='dbname'
Oracle LONGs (deprecated)	Oracle	Thin	oracle.jdbc.OracleDriver jdbc:oracle:thin:@'host':1521:'SID'
Oracle LONGs (deprecated)	Oracle	OCI	oracle.jdbc.OracleDriver jdbc:oracle:oci:@'TNSNAME'

Table 3.1: JDBC-Drivers

- **Home directory - avw.base.dir:** This is the root directory for all relative paths, if you leave it empty the current direcory of the start script is used.
- **Directory of form classes - avw.formclassdir:** Directory, where the system writes the form classes.
- **Directory for temporary files - Httpd.tempDir:** Directory for temporary files.

3.5 Logging

- **Log file - logger.logfile:** Name (path) of file, where **@enterprise** writes log information. If file not exists, a new one will be created. If the logfile extension is .zip or .gz, the logfile(s) will be compressed automatically depending on parameter *Restart log* or *Max. filesize*.
- **Restart log - logger.restart.logfile:** Here you can define how often the log file should be initialized - daily (at midnight) or at startup only.
- **Number of logs - logger.keep.logfile:** The number of stored log files. If the logfile extension is .zip or .gz, the logfile(s) will be compressed automatically.
- **Error file - logger.errorfile:** This file is a centralized collection of errors. Errors will also appear in the general logfile. If the logfile extension is .zip or .gz, the logfile(s) will be compressed automatically depending on parameter *Restart error log* or *Max. filesize*.
- **Restart error log - logger.restart.errorfile:** see *Restart log*
- **Number of error logs - logger.keep.errorfile:** see *Number of logs*
- **System loglevel - ep.logger.level:** This setting is used for log actions of **@enterprise** (applications) only. One of the following levels can be selected:

  Inherit If selected, level of parameter *root.logger.level* in section *Other parameters* is taken.

  ERROR Errors are logged only.

  WARN In addition to level *ERROR* warnings are logged.

  INFO HTTP requests are logged (time stamp, user, IP-address, and URL).

  DEBUG SQL-statements and process-oriented logging.

  TRACE The full HTTP-headers, parameters of prepared statements and other information for debugging purposes.

  Don't use the options DEBUG or TRACE in production for extended periods of time, because it generates a lot of data.

- **Store loglevel - store.logger.level:** Analog to *System loglevel*, but is used for log actions of **@enterprise** Store (package *com.groiss.store*) only. You can select the entry *Inherit* to use the selected level of *System loglevel*.

- **Servlet loglevel - httpd.logger.level:** Analog to *System loglevel*, but is used for log actions of **@enterprise** servlet methods (package *com.groiss.servlet* and Java Melody) only. You can select the entry *Inherit* to use the selected level of *System loglevel*.
- **Log on console - logger.logOnConsole:** The log information is written to the standard output stream.
- **Custom loglevels - logger.custom.level:** **@enterprise** has different loggers which can be customized here with a list separated by semicolons. It is possible to increase or decrease the trace level for a logger like in following example:

```
com.groiss.servlet.Dispatcher=ERROR;
com.groiss.store.impl.StoreEJB=DEBUG;
```

  All other loggers use the default settings.
- **Application loggers - logger.application.packages:** Define a comma separated list of application loggers as package/class/logger name, e.g. *com.groiss.itsm* means that all logging actions of this package are using the *System loglevel*.
- **Length of tail - logger.tail.length:** This parameter defines how much rows are displayed by default at the end of a log file via GUI (see *Admin tasks* → *Server* → *Logging*).
- **Max. filesize - logger.max.filesize:** This option can be specified in bytes, kilobytes, megabytes or gigabytes by suffixing a numeric value with KB, MB and respectively GB. For example, 5000000, 5000KB, 5MB and 2GB are all valid values.
- **Number of configuration backups - keep.conffiles:** This parameter defines how many backups of the configuration files (avw.conf and self-defined configuration files) should be kept. If saving the configuration via GUI, a backup file will be created in the appropriate folder where the configuration files exist.

3.6 Classes

- **Authorization Class - HttpdAuth.class:** **@enterprise** allows the usage of different authorization mechanisms. The Java class used is specified here. The default class (part of the distribution) is com.groiss.org.PasswdAuth.
  A special class is com.groiss.ldap.LDAPPasswdAuth which allows to authenticate against a LDAP server. The password check at login is delegated to those server. After setting this class and reloading the configuration navigation tree, a new section *User authorization via LDAP* will appear (more details are available in section 3.19).
- **Settings Class - settings.class:** A class defining some global settings can be defined here. For details see the **@enterprise** Programming Guide.

- **Notification Provider Class - avw.notification_provider_class:** The class for the notification mechanism, must implement the interface com.dec.avw.notification.NotificationKit. This mechanism allows to notify RMI-based Java clients in an asynchronous manner about changes in worklists.
- **Archiving Class - avw.archiveclass:** The class used for archiving process instances, must implement the interface com.groiss.wf.ProcessArchiver. If archiving should be prevented, configure com.groiss.wf.NoArchiver as archiving class!
- **Error-Formatter Class - avw.error.formatter:** You can write an error formatter class that will be used to display errors. The class must implement the com.groiss.gui.ErrorFormatter interface.
- **Services - services:** The list of services that the system starts. You can add your own services but should not modify or delete the entries already there, if you don't really know what you are doing.
- **Form class package - ep.form.class.package:** This parameter allows to define the default form class package for new created form types (existing form types keep their previous package name).

3.7 Localization

- **List of locales - Locale.list:** Here you can define a comma-separated list of locales that will be used by the server. If you don't define anything here, the server will use the following default locales: en_GB, en_US, de_DE, de_AT, and de_CH.
- **Language - Locale.language:** Defines the language for the user interface. Language is defined in ISO language code, for example de for German.
- **Country - Locale.country:** ISO country code, for example AT for Austria.
- **Variant - Locale.variant:** A default variant to use. You can define free variants in the list of locales (e.g., regions, companies, etc.).
- **Server timezone - avw.timezone:** This parameter allow to enforce a specific time-zone for all users. If nothing is selected, the default timezone of the server (operating system) is used, otherwise the selected one.
- **Decimal format - avw.decimal.format:** Define a decimal format as described in JAVA APIDoc.
- **Decimal separator - avw.decimal.separator:** Set the separator for floating-point numbers (default is .)
- **Decimal grouping separator - avw.decimal.grouping.separator:** A separator for e.g. thousand delimiter can be defined here (see Java APIDoc for more details).
- **Date format - DateFormat:** Format mask for date input and output. See the table 3.2 below for a description of the possible values.

- **Time format - TimeFormat:** Format mask for time input. See the table 3.2 below for a description of the possible values.
- **Default unit for displaying time intervals - calutil.defaultunit:** Default-Unit in seconds, minutes, hours, days and weeks.
- **Applet look and feel - applets.lookandfeel:** Specify the look-and-feel of the process editor, values are: metal or windows.
- **Max. table length - table.maxsize:** Specify a natural number. For tables of size greater than this number the user is asked before the table is shown.
- **Items per page - table.pagesize:** This defines the maximum number of entries in tables when paging is enabled. This parameter is also used for DOJO object selects.
- **Max. paging table length - table.paging.maxsize:** For paged tables of size greater than this number the user is asked before the table is shown or the search function in toolbar must be used.
- **Use browser language - locale.from.browser:** If this option is set, the system uses the language settings of the browser instead of the settings in the user table of **@enterprise**.
- **Always use server-timezone - use.server.timezone:** If this checkbox is set, the determined timezone of client Browser will be ignored and either the timezone set by the user (on user mask or user settings mask) or the *Server timezone* will be used.
- **Select list search option - selectlist.search:** The search option for searching in a select list:
  - Starts with: at the begin of a string
  - Substring: within a string
- **Enable Wiki link syntax - ep.wikilinks.enable:** If this checkbox is activated, links can be entered in the description of an ActivityInstance in wiki syntax: [[link | text]] or [[link]]
- **Default tab in process details:** Define a tab id as default when opening a process detail window. Default value is "admin.history". See the System Administration Manual, section Processes, for details (field "Detail tabs" in process definition mask).
- **Show toolbar in process-details popup:** When the process details are opened in a separate window, this checkbox enables a toolbar showing the possible actions for this process instance. For example, if you use the standard search for searching a process instance that is in your worklist, you will see the worklist actions in the search result details of this process.

Table 3.2 shows possible values for the date and time format masks.
The count of pattern letters determine the format.
(Text): 4 or more pattern letters–use full form, < 4–use short or abbreviated form if one exists.

Symbol	Meaning	Presentation	Example
G	era designator	(Text)	AD
y	year	(Number)	1996
M	month in year	(Text & Number)	July & 07
d	day in month	(Number)	10
h	hour in am/pm (1 12)	(Number)	12
H	hour in day (0 23)	(Number)	0
m	minute in hour	(Number)	30
s	second in minute	(Number)	55
S	millisecond	(Number)	978
E	day in week	(Text)	Tuesday
D	day in year	(Number)	189
F	day of week in month	(Number)	2 (2nd Wed in July)
w	week in year	(Number)	27
W	week in month	(Number)	2
a	am/pm marker	(Text)	PM
k	hour in day (1 24)	(Number)	24
K	hour in am/pm (0 11)	(Number)	0
z	time zone	(Text)	Pacific Standard Time
’	escape for text	(Delimiter)	
”	single quote	(Literal)	’

Table 3.2: Values for Date and Time Format Masks

(Number): the minimum number of digits. Shorter numbers are zero-padded to this amount. Year is handled specially; that is, if the count of ’y’ is 2, the year will be truncated to 2 digits.
(Text & Number): 3 or more–use text, less than 3–use number.
Any characters in the pattern that are not in the ranges of [’a’..’z’] and [’A’..’Z’] will be treated as quoted text. For instance, characters like ’:’, ’.’, ’ ’, ’#’ and ’@’ will appear in the resulting time text even if they are not embraced within single quotes.

3.8 Communication

- **SMTP host - mail.smtphost:** Server for outgoing emails (host name or IP address). It is also possible to define the smtp port in this field which must be separated by a colon, i.e. <smtp_host>:<smtp_port>.
- **Mail sender - mail.sender:** The mail address that will appear in the *from* field of mails that the system sends.
- **SMTP Username - ep.mail.smtp.username:** The user name for SMTP server (SMTP host).
- **SMTP Password - ep.mail.smtp.password:** The password of SMTP user.
- **Type of SMTP communication - ep.mail.smtp.communicationtype:** This setting is used by all communication possibilities in **@enterprise** for sending mails. One of the following communication types can be defined here:
  - *Unencrypted:* The content of the mail will be transferred without encryption. This is the standard communication type.

  - *STARTTLS:* This is an extension to plain text communication protocols, which offers a way to upgrade a plain text connection to an encrypted connection instead of using a separate port for encrypted communication.
  - *Encrypted:* The mail will be SSL–encrypted. The validity of the mail server certificate will not be checked.
  - *Trusted (with certificate):* To assure a secure transmission the mail server has to authenticate itself adverse **@enterprise**. This is achieved by checking the mail server certificate. To add a new certificate for a mail server it has to be added to the key store of **@enterprise**.

- **Administrator email address - avw.adminemail:** One ore more email addresses of the system administrator separated by comma. Beside this field the check function allows to test the SMTP settings by sending an email to administrator's email address.

- **Subject pattern - ep.mail.subjpattern:** An email subject consists of <subject> (<pattern> <pid>). In this field only the <pattern> part could be entered which is needed for identification, e.g. the text *ID:*. If subject pattern is entered, the email will be assigned to an existing process with given <pid> (if available). After this attempt the given *Action* of mailbox is executed.

- **Email notification text - ep.mail.notification.text:** Free text, which is the notification text (in email) for the user to inform him about new mail which has been added to the given process. This field allows to use following placeholders:

  %org% - the name of the organizational unit
  %proc_id% - the process instance id of the process where email is attached
  %task% - the current task of the process instance
  %from% - the sender of the email

- **Non trustworthy senders - ep.mail.junk:** The mails of all mailboxes will not be handled for given email addresse or a pattern of addresses. Separators are new lines. Examples:

  - *groiss* - If email address contains string "groiss", email will not be handled by mailbox
  - *groiss.com - If email address contains string "groiss.com" at the end, email will not be handled by mailbox
  - max.muster@* - If email address contains string "max.muster@" at the beginning, email will not be handled by mailbox
  - max.muster@groiss.com - If email address contains exactly the string "max.muster@groiss.com", email will not be handled by mailbox

- **Default action for sending mails - ep.mail.queue.usage:** Here you can define the default action for sending emails. Following values are available:

  - Send over mail queue: This option allows to send mails by using mail queue. If an error occurs, the mail will be stored in mail queue until *MailQueueTimer* is executed (see System Administration Guide - section *Timers* for more details).
  - Put in mail queue: If this option is selected, mails are stored in mail queue without sending attempt. The mails are sent when *MailQueueTimer* is executed (see System Administration Guide - section *Timers* for more details).
  - Send without mail queue: If this option is selected, mails will be sent immediately without using the mail queue. This is the default setting.

- **Max. time for mail queue item (minutes) - ep.mail.queue.maxtime.minutes:** The *MailQueueTimer* iterates over the mail queue and handle each entry which has a creation date. This creation date is used for this configuration parameter and if max. time is exceeded, the administrator will be informed and a appropriate status message will be written for this mail queue entry. The default value is 24 hours (= 1440 minutes).

- **SMTP default properties - ep.mail.smtp.defaultprops:** Define default properties for SMTP mail communication (see http://javamail.java.net/nonav/docs/api/). In particular the following properties are useful in dealing with network problems: *mail.smtp.connectiontimeout* and *mail.smtp.timeout*.

  Please note that the properties *mail.smtp.host* and *mail.smtp.port* cannot be overwritten by using this field, because the definition is done with **@enterprise** configuration parameter *SMTP host - mail.smtphost*!

- **IMAP default properties - ep.mail.imap.defaultprops:** Define default properties for IMAP mail communication (see http://javamail.java.net/nonav/docs/api/). In particular the following properties are useful in dealing with network problems: *mail.imap.connectiontimeout* and *mail.imap.timeout*.

- **POP3 default properties - ep.mail.pop3.defaultprops:** Define default properties for POP3 mail communication (see http://javamail.java.net/nonav/docs/api/). In particular the following properties are useful in dealing with network problems: *mail.pop3.connectiontimeout* and *mail.pop3.timeout*.

- **RMI port - avw.rmi.port:** Port number of RMI (Remote Method Invocation) listener. Needed for Java-Clients.

- **Enable RMI class loading - avw.dyn_class_load.enabled:** Enables class loading via RMI, this is needed when working with forms and the Java-Client.

- **Enable full RMI access - avw.rmi.enablefull:** When starting an RMI session the system must authorize a user. All RMI calls will be performed as this user then. If you enable full RMI access, you can call all available API methods independent of the user's rights.

- **Allow plain communication over RMI - avw.plain_rmi:** Allows plain (unencrypted) communication for RMI connections.

- **Export port for plain RMI communication - avw.rmi.plain_exportport:** If specified, this is the port used for RMI traffic.
- **Use SSL for login sequence at RMI communication - avw.secure_login_rmi:** Use SSL to encrypt the login sequence for RMI communication.
- **Crypt RMI communication with SSL - avw.permanent_secure_rmi:** Encrypt the whole communication over RMI.
- **Export port for RMI over SSL - avw.rmi.ssl_exportport:** If specified, the encrypted RMI communication uses this port.
- **Client certificates for RMI over SSL - ssl_requireclientcertificate_over_rmi:** This parameter determines how a secure SSL connection can be established by a RMI client. There are three possibilities:
  - **Are not requested:** If this option is selected, SSL connections are established in any case.
  - **Are required:** If this option is selected, SSL connections are established only if the client has a valid certificate for authorization.
  - **Are requested:** If this option is selected, the establishment of SSL connections depends on the content of the response: if the response contains a valid client certificate the SSL connection is established automatically; if the response contains no valid client certificate a login mask will be displayed to the user and after a successful login the SSL connection will be established.
- **Enable Wf-XML - avw.wfxml.enabled:** Defines, if this server is Wf-XML enabled. Possible values are *Off*, *Active*, or *Passive*. For further details on how to set up and use Wf-XML, please take a look at the section *Communication with other Systems* → *Wf-XML* of the **@enterprise** Application Development Guide.
- **WfXML OU - wfxml2.orgunit:** Default Wf-XML Organizational Unit.
- **WfXML User - wfxml2.user:** Default Wf-XML User.
- **WfXML Server - wfxml2.server:** Defines the default Wf-XML server.
- **WfXML access log for - wfxml2.log.objects:** Defines the objects, which will be logged. You can select between
  - ServiceRegistry
  - Factory
  - Instance
  - Activity
  - Observer.
- **Size of log - wfxml2.log.size:** Max. size of the logfile.
- **XMPP server address - xmpp.server:** The location of the XMPP-Server.

- **XMPP server port - xmpp.port:** Definition of port to XMPP-Server.
- **Application Repository URLs - appl.repository.urls:** A comma separated list of URL's for application repositories can be defined here. With these URLs **@enterprise** checks periodically for new versions (of **@enterprise**) and (if defined) for installed applications. For more details see section 4.4.

Hint: To enable RMI communication you must at least enable either *Allow plain communication over RMI*, *Use SSL for login sequence at RMI communication*, or *Crypt RMI communication with SSL*.
If a timer doesn't catch an exception, **@enterprise** sends a mail to the system administrator and deactivates the timer.

3.9 Cluster

See section 5.3.3 in the chapter about clusters for details about configuring clusters.

3.10 Workflow

- **Open form on process start - avw.start.with.form:** In the process start mask there is a checkbox where the user can decide to see the process form immediately after process start. Here you can define the default value of this checkbox.
- **Inherit Ids to subprocesses - avw.inherit.ids:** Don't create Ids for subprocesses - use the parent processes' Ids instead.
- **Enable application-spanning process definition - avw.procdef.appl_spanning:** If this option is set active (by default), it is possible to define processes with application-spanning elements (i.e. Forms, Tasks, Subprocesses and Roles as Agents). If this checkbox is not checked, elements of the current application are available for process definition only.
- **Allow automatic take - avw.autotake:** Allows users to take tasks automatically if they perform a function directly on an entry in the role-worklist or suspension worklist. This will only work if you add additional functions to the GUI of these worklists (e.g., the finish function). If the process-form of such a task is edited, the current editor is written in table *avw_currenteditor* and is visible in the process-instance history.
- **Show choice selection when single path - ep.choice.showsingle:** If this checkbox is not activated, no choice-mask is displayed anymore when one branch of a choice-object is active only. If activated, the choice-mask is displayed always.

3.11 DMS

- **Versioning - avw.dms.versioning_strategy:** *Not automatically* disables automatic version creation. *On agent change* creates a version if a different user edits the document (so, if the same user edits a document multiple times, no documents are created).

On every change creates a version every time the document is edited - an exception is the adaption via WebDAV client (e.g. MS Word): if a document is opened and stored multiple times, only one version is created (= for each "session" only one version is created).

- **Inherit permission list - avw.dms.bequest_acl:** When this option is checked, the permission lists of a folder is inherited to the contents of the folder.
- **DMS Storage Class - IStore.class:** You can specify your own DMS storage class here. The class must implement the interface com.groiss.dms.IStore.
- **DMS Archiving Class - DMSArchiver.class:** Class for archiving documents, must implement the interface com.groiss.dms.DMSArchiver.
- **Standard table model / Table handler - avw.dms.standard_tablemodel:** A class can be specified, which is used for displaying the document tables. For further details please take a look into **@enterprise** Application Development Guide, section *Using the DMS API*.
- **WebDAV drive - webdav.drive:** The webdav drive can be specified with this parameter, which represents the root (the same letter like set in WebDrive properties). If the value *off* is entered, WebDrive will not be used anymore. You have to reconnect to the **@enterprise** Server after changing this parameter.
- **Open documents in new window - avw.dms.newwindow:** If checked, documents will be opened in new window.
- **Open documents via Microsoft Office Plug-in - webdav.officeDocuments.openWithPlugin**: This must be checked to activate usage of the plug-in by **@enterprise**. More information can be found in section 3.11.1.
- **Extensions of plug-in supported documents - webdav.officeDocuments.openWithPlugin.extensions**: Holds a comma separated list of extensions (e.g. doc, docx) for which the Microsoft Office Plugin should be used. More information can be found in section 3.11.1.
- **Open text files via text editor:** This must be checked to activate usage of the text editor in the **@enterprise**.
- **Extensions of text editor supported files:** Holds a comma separated list of extensions for which the text editor should be used. Only for DMS documents which extension is contained in that list the editor will be used. Supported document types are: txt, asc, csv, etx, rtx, tsv, wml, wmls, xml, htc, css.
- **Use applet for data capturing - ep.dms.use.applet:** When creating a new document in DMS you may use an installed webcam or scanner as source for your document's content. This is done via an applet which must be activated with this checkbox. If that option is activated, the document creation dialog will provide the option to select any device for which a TWAIN driver is installed.

- **Maximum document size (in bytes) - avw.dms.max_doc_size:** You can define a maximum size for DMS documents here. **@enterprise** will not allow users to create documents that are bigger than this value. If you don't define a maximum size, there will be no size restriction for DMS documents. Anyway, also databases can limit the maximum size.
- **Character set for text files - avw.dms.textfile_charset:** Here you can enter the character set for text files, if the content of these files is not displayed correctly, e.g. the content of the file has ANSI charset, but the server charset is UTF-8 - for this purpose set the character set for text files to the value *CP1252* (if client is running under Windows only).
- **Full-text search - avw.dms.textsearch.state:** With the help of this parameter the state of the full–text search can be determined: There are three possible states:
  - **Switched off:** No full–text search is used at all.
  - **String search in form fields:** The database doesn't support full–text search. Therefore the required string can be searched in a table containing all string values of form fields.
  - **Activated:** The full–text search of the current database is used.
- **Check permissions on DMS folder content - dms.check.rights.on.list:** If activated, the view-right is checked when folder content is read.
- **Use Recycle Bin - ep.dms.use.recyclebin:** If this checkbox is activated, the recycle bin for DMS objects is used, i.e. if a DMS object is deleted, it will be moved to recycle bin first and will not be deleted. More information concerning this topic is available in the **@enterprise** user manual.
- **Show extensions - avw.dms.showextensions:** Show the document name extension, e.g., .doc or .txt.
- **Do not display hidden documents - avw.dms.hide_hidden_docs:** If this option is checked, users cannot see any hidden documents (beginning with a point in the filename) in the DMS.
- **Do not display folder Common - dms.hide.common:** With this parameters you can hide the *Common* folder in DMS.
- **Do not display user related folders - dms.hide.userfolder:** This parameter fades out the user folder and the folder of the substituted person
- **Sort table grouped by folders/documents - dms.grid.sort.grouped:** If this parameter is checked the elements of a DMS folder will be grouped into folders and non-folders and sorting (e.g. by name) will be performed within this groups (as it is known by Windows Explorer). This parameter works in smartclient only!
- **OpenOffice home - ep.openoffice.path:** The root path of OpenOffice can be entered here for replacement of Office templates (odt-files). More details about Office templates in **@enterprise** can be found in the *Application Development Guide*. Please

note that a mixture of 32-bit and 64-bit version of JAVA and OpenOffice can lead in problems (e.g. converting mechanism from odt to another file-format cannot be used)!

- **Number of named OpenOffice pipes - ep.openoffice.threads:** Here you can enter the number of threads which are used for the piped connection with OpenOffice. The default value is 1.
- **Use named pipes to connect to OpenOffice - ep.openoffice.piped.connection:** If this checkbox is activated, a piped connection will be established with OpenOffice. This option is activated by default.
- **Used ports for OpenOffice connection - ep.openoffice.ports:** Alternative to named pipes connection a connection via ports (socket connection) can be used. For this purpose a port (or a comma separated list of ports) must be entered and the checkbox *Use named pipes to connect to OpenOffice* must be deactivated. The default port is 210.
- **Support conversion to 'Office Open XML' types (docx, xlsx, etc.) - ep.openoffice.support.office.open.xml:** Activate this checkbox, if function *Generate document* in DMS is used to create Office Open XML types. This parameter is inactive by default, because LibreOffice does support these types only!
- **Files ignored in zip upload - ep.dms.zipupload.ignore:** Enter a comma separated list of path names which are ignored by DMS function *Zip upload*. Allowed are also * and ? as wildcards. The whole path is always compared (case-sensitive), i.e. if you want to hide directory .xx and its content, you have to enter *.xx**

3.11.1 Edit Microsoft Office Documents via Browser

Via WebDAV it is possible to open Microsoft Office Documents in read-write mode when clicking on a document link in the browser. Therefore the new **@enterprise** GUI will use the browser plug-in that will be automatically installed if Microsoft Office is installed on your computer. For using that feature two aspects must be configured:

1. activate the usage of the plug-in
2. configure user authentication

Plug-in activation: Therefore section 3.11 of the configuration provides the following parameters:

- **Open documents via Microsoft Office Plug-in - webdav.officeDocuments.openWithPlugin**: This must be checked to activate usage of the plug-in by **@enterprise**.
- **Extensions of plug-in supported documents - webdav.officeDocuments.openWithPlugin.extensions**: Holds a comma separated list of extensions (e.g. doc, docx) for which the Microsoft Office Plugin should be used. Only for DMS documents which extension is contained in that list the plug-in will be used - all other documents will be handled with the browser's default behavior.

Note: It may be the case that the plug-in is deactivated by the browser itself. In that case it must be activated within the browser application (e.g. via Add-Ons/Plugins in FireFox). If a Browser (e.g. Google Chrome) does not support the plug-in, the MS Office URI schemes are used to open the MS Office document in an editable way!

Authentication: For viewing/editing a DMS document we must determine the requesting user to check if he is permitted for that action. But the session cookie of the browser cannot be passed to the plug-in therefore we need alternative ways for authentication of the requesting user. The following parameters in section 3.14 of the configuration are determined to control the various ways for authentication:

- **Basic-Auth in WebDAV - avw.dms.allow_basicauth**: When activated the server will send an authentication request to the client if no user could be determined. The client will react by opening a dialog to enter the user's id and password that will then be sent to the server.

  Usage of Basic Authentication may be deactivated on your windows client but you may change the current behavior by setting registry entry HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters\BasicAuthLevel.

  Supported values are:

  - 0 - Basic authentication disabled
  - 1 - Basic authentication enabled for SSL shares only
  - 2 or greater - Basic authentication enabled for SSL shares and for non-SSL shares

  When setting value to 2 be aware that username/password will be transmitted in clear text when using a non-SSL connection.

  Note: Activating Basic-Auth in **@enterprise** is not only relevant for editing documents via the browser but for all kinds of WebDAV clients trying to connect to the DMS of **@enterprise**.

- **Use persistent cookie in DMS - ep.dms.useperscookie**: When checking this parameter a persistent session cookie will be written on the client when the user navigates within the DMS in the browser. That cookie will then be sent by Microsoft Office to authenticate the user. But note: this works only when using Internet Explorer as browser, otherwise Office will not find that cookie.

- **Use authentication token in DMS - ep.dms.useauthtoken**: When checking this parameter an authentication token will be passed as part of the URL to Microsoft Office. The server will be able to authenticate the user by this token.

- **WebDAV authentification class - webdav.auth.class**: Here you may specify an implementation of interface com.groiss.servlet.WebDAVAuth which will be used to

determine the requesting user by some data in the HTTP request (e.g. from a client certificate sent as part of that request).

As for Basic-Auth this setting will be used for all kinds of WebDAV clients.

3.12 Search

- **Maximum table size on server (rows) - query.maxtable:** Maximum table size the server will handle. If the table size exceeds this value, the operation is canceled and an error message is produced.
- **Cache interval (minutes) - monitoring.cacheinterval:** Specifies, how long a query result resides in cache.
- **Maximum number of cached queries - monitoring.cachesize:** Number of queries in cache.
- **Maximum number of simultaneous queries - monitoring.maxparallel:** Number of threads, that concurrently compute query results.
- **Maximum number of startable queries - monitoring.maxqueue:** Length of queue of queries waiting for execution (waiting for a free thread).
- **Default process Id search type - avw.reporting.defaultIdSearch:** Here you can define the standard type for id search in *Process Search* - see user manual for further information.
- **Default subject search type - avw.reporting.defaultSubjectSearch:** The same as *Default process-id search type*, but for subject.
- **Process relations - avw.process.relations:** It is possible to define a relationship between process instances. The relation is defined as *ProcessRelation(ProcessInstance p1, ProcessInstance p2, String reltype)*. The relation can be maintained via API or with the task-function *addRelation*. The available relation types can be defined in the field *Process relations*. For each relation type a pair of id and name1/name2 is defined, names and id separated by whitespace (see syntax beneath). A comma, new line feed or carriage return separates the pairs. The id is stored in the database relation, the names are used in the user interface.

  Definition syntax:

  id [name1 [name2]]{sep id [name1 [name2]]}

  If name2 is missing, name1 is the default. If name1 is missing, the id is the default. In name1 and name2 it is possible to use %20 to use blanks in names (values). The names could be internationalized by adding @@@appid:key@@ (appid is the ID of the application; *ep* is **@enterprise** default).

  Examples:

```
a
b B
c C%201 C2
d @@@ep:process@@
e @@@ep:forward@@ @@@ep:back@@
```

- **Search case-insensitive by default - avw.reporting.defaultIgnoreCaseSearch:** If this checkbox is activated, the checkbox *Ignore Case* on process search mask is activated by default.
- **Exact Id short search only - avw.reporting.exactIdShortSearch:** If this checkbox is activated, you have to enter the right Id to get a correct result.
- **Short search includes subject - avw.reporting.shortSearchSubject:** If this checkbox is activated, the subject will be included in shortsearch.
- **Short search includes field values - avw.reporting.shortSearchFieldvals:** If this checkbox is activated, fieldvals will be included in shortsearch. It is necessary that full-text search is activated and the checkbox *Useable in DMS* must be activated for each formtype which should be found.
- **Order process Ids by OID - monitoring.orderProcessId:** In worklist and Reporting processes will be sorted by OID, if this checkbox is activated.

  For more information on process relations read the corresponding chapter of the **@enterprise** Application Development Guide.
- **Show all rows, even when no view right - avw.reporting.showNoAcces:** If this checkbox is activated and the user who uses search-engine has no view right on DMS-object, he will get all rows as result.
- **Use underscore (_) as SQL wildcard - avw.reporting.underscoreIsWildcard:** If this checkbox is activated, underscores are allowed as SQL wildcard. If activated, it is not possible to search for '_' unless you escape it yourself.
- **Use smart search algorithm for multi-field searches - list.smartsearch:** If activated, it will be searched globally in all specified fields of a table by using OR-joins. This parameter takes effect on
  - short search in select-list and select-table
  - DOJO object select
  - short search in object-table (e.g. **@enterprise** administration master data tables)
  - short search in form-table
  - short search in select-list of function *Change Agent*

  Example: The search fields are firstname and surname of the user table. In short search field of the user table the string *Roland Eisenberg* has been entered. The sql-condition would be following:

```
(lower(firstname) like(Roland%) or lower(surname) like(Roland%))
and (lower(firstname) like(Eisenberg%) or lower(surname) like(Eisenberg%))
```

It is also possible to activate/deactivate this behavior for each table by setting following attribute in configuration file (e.g. myappl.xml):

```
<Attrib key="smartSearch" value="true|false"/>
```

- **Show time in date conditions - avw.reporting.showTimeInDateConditions:** If activated, date and time for datefields on process-/document-searchmask and Reporting condition mask can be entered and on all these masks the appropriate checkbox is activated by default. If this checkbox is deactivated, only the date can be entered as value (without time).
- **Open forms in edit mode - avw.reporting.openFormLinksInEditMode:** Activate this checkbox, if forms in reporting result should be opened in edit mode.
- **Search-delay for ObjectSelects (ms) - objectselect.search.delay:** This delay is used in ep/widget/ObjectSelect, if a value is entered. If the delay-period is over, the entered value as yet is sent to the server.

3.13 Tuning

With the following parameters the system's performance can be influenced.

- **Worklist Cache - avw.wlcache.state:** Specify, whether the worklist cache should be used. *Activated* means that the cache is used; *Started (but not active)* means that data structures are maintained, but the cache is not used for worklist construction; *Switched off* means that the cache is not used and data structures are not maintained.
- **Do not cache seen objects - avw.wlcache.exemptseenobjects:** If this checkbox is activated, seen objetcs will not been cached anymore and read from database.
- **Do not cache user folders - avw.wlcache.exemptuserfolders:** If this checkbox is activated, user folders of personal worklist will not been cached anymore and read from database.
- **Defer loading of finished parents - avw.wlcache.parents.defer.missing:** If checked, loading of missing parents (for finished parfors, scopes, processes) can be deferred.
- **Throw Exceptions when Worklist Cache is not available - avw.wlcache.unavailable.restrict:** If checked, then queries to an incative worklist cache (e.g. during startup) will not fall back to database queries but throw an exception instead. This avoids database overload during cache startup. Recommended for large installations with millions of active activity instances where worklist cache startup may take several minutes.

- **Reload classes - avw.class.reload:** Reloads classes without server restart if possible. This should be used only in development environments.
- **Statement statistics - avw.stmt.statistics:** Creates statistics of database statements. If enabled, you will see how often statements have been executed and how much time they consumed (total and average). You can find these statistical information in *Admin-Tasks → Database connections* . Don't activate statement statistics for long time periods in production environments because they may need a lot of resources and therefore slow down your server.
- **File cache size (in megabytes) - file.cache.size:** Here you can define the size of the web-server file cache. The default value is 20MB.
- **Permission Cache activated - aclcache.active:** Check, if the ACLCache should be activated. More details about ACLCache can be found in section 3.13.1.
- **Max. number of object specific rights - aclcache.objectrights.maxelems:** Size of the object specific rights cache (in objects).
- **Lifetime of object specific rights (sec.) - aclcache.objectrights.lifespan.secs:** Lifetime of rights in the object specific rights cache.
- **Max. number of class rights - aclcache.classrights.maxelems:** Size of the class rights cache (in objects).
- **Lifetime of class rights (sec.) - aclcache.classrights.lifespan.secs:** Lifetime of rights in the class rights cache.
- **ACLCache parameter:** See section 3.13.1
- **Monitor server with Java Melody - ep.servermonitor.use.melody:** If activated, Java Melody is used as server monitor in **@enterprise**.
- **Monitor DB Connections with Java Melody - ep.dbmonitor.use.melody:** If this parameter is checked, the database connections will be monitored and displayed in **@enterprise** Servermonitor. This option is usable, if parameter *Monitor server with Java Melody* has been activated before!
- **Use CompressionFilter in Servlets - ep.servlet.use.compression:** If activated, the compression filter is used in **@enterprise**. This filter can, based on HTTP headers in a HttpServletRequest, compress data written to the HttpServletResponse, or decompress data read from the request. When supported by the client browser, this can potentially greatly reduce the number of bytes written across the network from and to the client.
- **Activate Axis Servlet - ep.servlet.use.axis:** If activated, the AXIS2 component in **@enterprise** is used. More details for using web services can be found in **@enterprise** application development guide, chapter *Web services*.
- **Enable Process Debugger - ep.process.debugger.enabled:** If activated, the Process Debugger component of **@enterprise** can be used. This component is described in **@enterprise** administration guide, section *Test cases*.

- **Inline images into CSS - ep.css.inline.images:** Inlining images (as Base64-strings) causes fewer server requests, but the CSS-file is significantly larger. It depends on your client/network configuration which option has fewer disadvantages.
- **Inline imported CSS-files - ep.css.inline.styles:** Inlining imported css-files causes fewer server requests, but the single CSS-file is significantly larger.
- **Allow automatic move into user folders - ep.userfolder.allow.automove:** If this checkbox is activated, worklist entries are moved automatically to appropriate user folders which have entered a XPath expression.
- **No initial listing for the following tables - ep.big.tables:** A comma-separated list of table id's can be entered here where no listing should be performed when displaying table the first time. It is necessary to perform the search function(s) of a table to list the content. Example configuration:
  The string admin_tree.user,admin_tree.dept means that no content is listed initially for user table and organizational unit table.

3.13.1 ACLCache

In **@enterprise** it is possible to speed up the rights check by activating the ACLCache. The cache improves the speed of the ACL.hasRight() method calls. The results of calls to method ACL.hasRight() are cached, and the cache is consulted before accessing the database. The cache is organized as an expirable and size bounded LRU cache.

The items have a maximum lifespan associated with them. If an item has been found in the cache, but has expired its lifespan, it is removed from the cache and is reported as being not in the cache. This behavior ensures, that cached right checks do not become unduly outdated. The value lifespan is configurable whereas the default value is 5 minutes.

The cache has also a maximum number of cached elements associated with it. If this number would be exceeded by the insertion of a new cached item, the least recently used item is removed from the cache, thereby ensuring a size bound while providing good hit rate.

Actually, there are two caches, one which stores acl-entries for specific objects and one which stores acl-entries for classes. The parameters for size and lifespan can be configured separately for those two caches.

- **Use partition optimized query for permission checks - acl.separate.targetquery:** ACL evaluations can be tuned by using separate queries for objectscope = 3 versus objectscope <> 3. For this purpose activate this parameter.
- **ACL list: Permission Cache integration - acl.list.cache.usage:** This parameter allows to define how ACL list interacts with ACLCache. Following options are available:
  - None: List does not interact with cache
  - Check only: Cache is consulted, no results are inserted into cache

  - Insert positive results only: Cache is consulted, only positive results are inserted
  - Full: Cache is consulted, all results (positive and negative) are inserted into cache)

- **ACL list: Max. number of OIDs in IN-Clause - acl.list.target.splitsize:** The split size for the target set of an ACL.list-query. If the size of the target set is not greater than the split size, **@enterprise** can filter by using a SQL IN-Clause with the target oid's, otherwise a more general filter will be used which may result in a larger result set for that query. In both cases a single SQL statement will be executed. Please note that there are database specific restrictions concerning the number of literals within an IN-Clause and also the textual length of an SQL statement.
- **ACL list: Always restrict by OID for the following classes - acl.list.target.splitclasses:** A comma-separated list of fully qualified class names. For those classes, the target set should be splitted so that more than one SQL statement will be executed which always filter by target oid's using an IN-Clause. This is useful if a lot of object specific permissions exists for such a class so that the more general filter would cause a huge result set.

3.14 Security

- **KeyStore file - ssl.keystore:** The Java KeyStore is a binary file, which holds the keys and certificates of the system and the certificates of trusted organizations, so called trust anchors. The KeyStore is the central "database" for certificate management. Ensure that there exists a backup of the KeyStore of **@enterprise**.
- **KeyStore password - ssl.keystore_pwd:** To access a KeyStore a password (with a minimum length of 6 characters) is needed.
- **Default validity period of certificates (days) - cert.default.validity:** If a self-signed certificate should be created, this value is taken for validity period by default.
- **Certificate alias to use for SSL connections - ssl.cert.alias:** Since each user can define his own certificate which is stored in the server keystore, the certificate alias to use for ssl connections has to be configured.
- **Password for server certificate - prk.passwd:** The Java API to access the KeyStore is not able to handle different keys with different key passwords. So a system key password has to be configured to access the keys. This password has a minimum length of 6 characters.
- **Bind session to IP address - ep.check.ip:** If activated, the real client ip address is checked with the ip address stored in session.
- **Basic-Auth in WebDAV - avw.dms.allow_basicauth:** Check, if you want to allow Basic-Auth authentication in WebDAV. More information can be found in section 3.11.1

- **Use persistent cookie in DMS - ep.dms.useperscookie:** Check, the persistent session cookie should be used for WebDAV access. . More information can be found in section 3.11.1
- **Use authentication token in DMS - ep.dms.useauthtoken**: Check, if an authentication token should be passed to Microsoft Office Plug-in. More information can be found in section 3.11.1
- **WebDAV authentification class - webdav.auth.class**: Here you may specify an implementation of interface *com.groiss.servlet.WebDAVAuth*. More information can be found in section 3.11.1.
- **Check Referer header - ep.check.http.referer:** When the referer check is enabled, the Dispatcher does not permit requests if the 'Referer' header is missing from the HTTP request, or when it does not match the request. A 'Referer' header matches the request, if the base part of the request URL (consisting of protocol/scheme, host, port and context-root) is a prefix of the 'Referer'.
  Methods and classes marked as public (via interface *com.groiss.servlet.Public* or annotation *com.groiss.servlet.Access.mode.Public*), are never subject to the referer check.
- **Exemptions from Referer check - ep.check.http.referer.exempt:** Additional exemptions from referer check can be configured here. A basic set of such method and class names is stated as default value of the configuration parameter. Removal of elements from this default list is not recommended, it must be done with great care to avoid application lock out!

3.15 Password policy

The parameters in this section are separable in 3 main groups, which are explained in the following paragraphs.
Note: No parameter of these groups is needed to be set, quite the contrary is recommended. If a too strict password policy is established - especially with the parameters of group 2 -, a brute-force attack may be effective in a small amount of time, because of the insufficient number of possible passwords.
So, if you don't want to set a parameter let the input field blank.

3.15.1 General Policy Settings

The following parameters do not focus on the password itself but on the password change- and login-management. These parameters are:

- **Period of validity (in days) - passwdpolicy.days_password_valid:** Defines the password's period of validity in days.
- **Inform user before password expires (in days) - passwdpolicy.days_warning_before:** Defines the days before the validation time is expired where the user will get a warning, that his password will expire.

- **Maximal number of unsuccessful logins until account is deactivated - passwdpolicy.max_count_invalid_logins:** A unsuccessful login is defined as a login attempt of an existing user id with a non valid password. If the specified number of unsuccessful logins are performed between two valid sessions of the specific user, the account is deactivated and the user will get a specific error message on the next login.
- **One-way Hash Algorithm to Use - passwdpolicy.algorithm:** The password is stored in encrypted form by using a one-way-hash function. In former releases this algorithm was the Unix Crypt algorithm. Now one of the following different algorithms can be chosen.
  - **SHA-256 (Secure Hash Algorithm):** Takes a plain string of any length and produces a 256-bit hash output. SHA is said to be secure and is the default value if nothing is configured.
  - **Unix Crypt:** Is limited to 8 bytes input (that means 8 characters), so it is not recommended to use Unix Crypt furthermore. Nevertheless, to ensure compatibility it is supported further on.
  - **SHA-1 (Secure Hash Algorithm):** Takes a plain string of any length and produces a 160-bit hash output. It is not recommended to use this algorithm anymore due to vulnerability!
  - **MD5 (Message Digest 5):** Takes a plain string of any length and produces a 128-bit hash output. MD5 is said to be secure and calculates the hash value faster than SHA.

3.15.2 Default Policy Checker Settings

The release is delivered with a default password checker which ensures proper passwords and which is highly configurable. If you need extended configuration options, it is possible to implement a special password checker.
The following parameters of the default checker can be changed to specify the minimum requirements for a password. The default values are 0!

- **Minimal length of password - passwdpolicy.min_length:** Specifies the minimal length of a password. As an example, if the parameter is set to 8, the password "soccer" is not accepted, but "icehockey" is. (Recommended:4)
- **Maximal length of password - passwdpolicy.max_length:** Specifies the maximal length of a password. As an example, if the parameter is set to 8, the password "hello_its_me" is not accepted, but "hello" is. (Recommended:8)
- **Minimal number of letters in password - passwdpolicy.min_letters:** Specifies the minimal number of letters in the password. As an example, if the parameter is set to 1, the password "1234" is not accepted, but "a1234" is. (Recommended:1)
- **Minimal number of capital letters - passwdpolicy.min_capitals:** Specifies the minimal number of capital letters in the password. As an example, if the parameter is set to 1, the password "hello" is not accepted, but "Hello" is. (Recommended:1)

- **Minimal number of lowercase letters - passwdpolicy.min_lowercase:** Specifies the minimal number of lowercase letters in the password. As an example, if the parameter is set to 1, the password "HELLO" is not accepted, but "hELLO" is. (Recommended:1)
- **Minimal number of digits - passwdpolicy.min_digits:** Specifies the minimal number of digits in the password. As an example, if the parameter is set to 1, the password "Hello" is not accepted, but "Hello1" is. (Recommended:1)
- **Minimal number of special characters - passwdpolicy.min_others:** Specifies the minimal number of special characters in the password. Special characters are defined as any character which does not belong to any of the following character classes: uppercase characters, lowercase characters, digits, space characters. As an example, if the parameter is set to 1, the password "hello" is not accepted, but "hello*" is. (Recommended:0)
- **Minimal number of different characters - passwdpolicy.min_different_chars:** Specifies the minimal number of different characters in the password. As an example, if the parameter is set to 3, the password "aaaa2222" is not accepted, but "aabb2222" is. (Recommended:3)
- **Maximal sequence of same character - passwdpolicy.max_char_adjacent:** Specifies the maximal sequence of the same character in the password. As an example, if the parameter is set to 3, the password "aaaa" is not accepted, but "aaabaaa" is. (Recommended:2)
- **Maximal length of substring - passwdpolicy.max_sub_user_data:** Specifies the maximal length of any substring in the password, which exists in the user's first name, surname or id. As an example, if the parameter is set to 3 and the user's id is "testuser" , the password "stus" is not accepted, but "tes ser" is. This check is case insensitive. (Recommended:2)
- **Number of old passwords to check - passwdpolicy.history_steps:** Specifies the number of old password to check password reuse. As an example, if the parameter is set to 3 and the user changed his password in the order "hello", "itsMe", "myPassword" and "letMeIn", the password "itsMe" is not accepted, but "hello" is. (Recommended:5)

  Note: The history check can only cover old passwords, which have been logged in the database. These old passwords are deleted by the LogTask Timer, so if the timer has deleted all old passwords according to his configuration, the history check can't be performed correctly. The result will indicate a correct password, although the password may have been reused and even be equal to the previous.
- **Minimal number of whitespace characters - passwdpolicy.min_whitespace:** Specifies the number of min. allowed whitespace characters.

Note: If parameters are set in a way that an inconsistent policy is specified, the users may not be able to change their passwords. So please care about the following rules for the parameters:

maximal length >= minimal length

minimum capitals + minimum lowercase characters + minimum digits +
minimal special characters <= maximal length

minimum letters + minimum digits +
minimal special characters <= maximal length

minimum different characters <= maximal length

3.15.3 Your Own Checker Class

- **Checker Class - passwdpolicy.checker_class:** If the default password checker does not satisfy your requirements, you can enter your own password checker class here. The class must implement the com.groiss.passwd.Checker interface.

3.16 Calendar

- **Holiday Class - avw.calendar.class:** Here you can define a class for displaying the holidays in the calendar. It must implement the com.groiss.cal.Holidays interface.
- **iMIP - calendar.imip:** If this checkbox is activated, iMIP will be used. In **@enterprise** calendar notifications contain *iCalendar*-files. iMIP offers the possibility to process status information of an appointment.
- **iMIP email address - calendar.imip.email:** Email-address which is used for communicate with the participants; participants will reply to this email-address!
- **Show default resource - cal.show.defaultres:** If this checkbox is checked the user can use a simple resource form for assigning resources to calendar appointments.
- **Resource classes - cal.resources:** It is possible to use arbitrary *Persistent* classes for calendar resources. Either the names (incl. package name) of the classes can be entered (e.g. a form class - com.dec.avw.appl.<formid>_<formversion>) or xml nodes can be defined in following way: xmlid:<xml_id>.<node_id>. The type of the xml node should be a *query* node (see Application Development Guide for more details). After setting the classes the self defined resources can be used in the calendar.
- **Non working day - cal.nonworkingdays:** In this list it is possible to select one or more non working days, which will be needed for example in escalations.
- **Calendar Class - calendar.class:** A calendar class of type com.ibm.icu.util.Calendar can be entered here which is used by **@enterprise**.
- **Number of days in agenda-view - calendar.agenda.days:** Define the number of days which are displayed in agenda view (how much days in advance).
- **Start of worktime - cal.worktime.start:** Definition of the time where worktime begins (needed for calculation of process plans).

- **End of worktime - cal.worktime.end:** Definition of the time where worktime ends (needed for calculation of process plans).
- **Worktime per day - cal.worktime.per.day:** Definition of worktime hours (needed for calculation of process plans). This value can be different from time period of start/end worktime. Example: start of worktime is 08:00, end of worktime is 17:00 and worktime per day has value 8.5. The difference between start and end is 9 hours, but the worktime per day is 8.5 hours only (0.5h is the lunch break for example). The calculation in plan tab considers this situation.
- **Calendar sources - cal.applications:** A list of classes can be defined here to activate/deactivate additional calendar-components, e.g. if *com.groiss.calendar.CalendarAppl* is removed, no appointments can be added anymore (default: com.groiss.calendar.CalendarAppl, com.groiss.calendar.wf.DueTasks, com.groiss.calendar.wf.FinishedTasks).
- **Date import formats - cal.impex.formats:** A list of classes which implements the com.groiss.cal.CalFormat interface. These classes are used for importing/exporting.
- **Notifier class - cal.notifier:** A class which implements the com.groiss.cal.Notifier interface. This class is used for sending notifications (reminder) of due appointments.

3.17 Process cockpit

This section contains the parameters for the process cockpit (see details in the *User Manual*).

- **Root folder - ep.cockpit.rootfolder:** The path to the root folder of the process cockpit.
- **Reports for all processes - ep.cockpit.allreports:** A comma separated list of Report-Id's. These reports are displayed in the process cockpit for every process.
- **Show overdue processes of last n days - ep.cockpit.deadline.days:** Specifies the number of days, which are used for the calculation of process deadline violations per process definition.
- **Show last n instances - ep.cockpit.recent:** Specifies the number of instances, which are displayed in tab *Runtime* of table *Recently Started* in process cockpit.
- **Common processes - ep.cockpit.commonproc:** A comma separated list of formtypes (Formtype-Id + Version, e.g. jobform_1) which contain the formfield *area*. The forms are used to assign process instances of common processes (for example project) to a cockpit entry.

3.18 Other parameters

In this area a various number of helpful parameters are listed. Each parameter has a short description which can be displayed by activating the help icon.

3.19 User authorization via LDAP

This section appears only, if the authorization class com.groiss.ldap.LDAPPasswdAuth has been set in section 3.6. It implements authentication against a directory server. The password check at login is delegated to this server. The *sysadm* user account is exempted from this, it will always be authenticated locally.
There are two general aspects to configured, namely the technical details of the connection to the LDAP server and the organizational details of the mapping of the **@enterprise** user ids to the LDAP entries. For the communication details, the following items have to be entered:

- **LDAP Host:** The hostname or IP address of the LDAP server.
- **Port:** The port of the LDAP host (default port is 389 for Unencrypted/STARTTLS and 636 for Encrypted).
- **Type of communication:**
  - *Unencrypted:* No encryption used. **Beware**: passwords are transmitted in plain. Recommended only for test environments.
  - *Encrypted:* Standard SSL/TLS encryption. No plain password transfer takes place.
  - *STARTTLS:* Start with plain connection and upgrade it to a secure one. No plain password transfer takes place.
- **Trust level:** Depending on selected communication type one of following trust level must be selected:
  - *System default:* The standard trust mechanisms of Java is being used, this is appropriate when the certificate of the directory server is an official one. Recommended for production use.
  - *Blind:* No real check of server certificate, no check of hostname. While blind trust may be fine for development environments, it is strongly discouraged to use it in a production environment.
  - *Certificate in truststore:* The **@enterprise** truststore is being used: if the server certificate has been imported there, it is trusted, even if it is a self signed one.
- **Timeout (ms):** Timeout in milliseconds for communication with the LDAP server. An empty value means no timeout at all.

For the organizational details of the mapping, two general cases can be distinguished:

- **Simple and Flat:** There is a single root entry your LDAP server to which all the relevant user entries are attached directly. The connection to the LDAP server is initiated with the (presumed) credentials of the user trying to log in. In order to authenticate against such a scheme, just the search path and the user id pattern have to be entered. For such a scheme, leave the *User* and *Password* configuration entries empty and enter the following parameters:

  - Search path: The location in the LDAP tree where the initial connection shoud be made. Usually the last part of the user pattern, but your mileage may differ, e.g.: ou=Development,dc=acme,dc=org
  - Pattern for User DN: A pattern which leads to the full DN of the LDAP user entries. The placeholder *${ep_uid}* which is substituted with the id of the **@enterprise** user, e.g.: cn=${ep_uid},ou=Development,dc=acme,dc=org

- **Dispersed or Hierarchical**: There are many different locations where user entries can be found in your LDAP server tree. Such a scheme requires authentication in three stages, namely to first connect with an administrative LDAP user (and her password), to search for an appropriate LDAP entry matching the user id and to rebind the connection with full DN credentials of the found user. For such a scheme, enter all of the following parameters:

  - User: DN of an (LDAP) user allowed to search for the entries, e.g.: cn=Manager,dc=acme,dc=org
  - Password: Password of this user.
  - Search path: The root of the part of the LDAP tree where user entries should be searched, e.g.: ou=Development,dc=acme,dc=org
  - Pattern for User DN: An LDAP filter expression which allows to search for the appropriate user entry. The placeholder *${ep_uid}* is substituted with the id of the **@enterprise** user trying to authenticate, e.g. (&(cn=${ep_uid})(objectClass=inetOrgPerson))

Conditional login according to LDAP group membership can also be accomplished. Let us assume that the organization or the user entries in your LDAP server are dispersed or hierarchical (see above), and that just a subset of all those users entries in the LDAP server are relevant as **@enterprise** account. A common form for such an organization would be to create an entry with object class *groupOfNames* or *groupOfUniqueNames* with e.g. a DN of *cn=epusers,ou=Development,dc=acme,dc=org* that represents the grouping and to add the full DN of the user entries as values of the *member* or *uniqueMember* attributes of the group entry. Then a pattern like
(&(cn=${ep_uid})(objectClass=inetOrgPerson)
(memberOf=cn=epusers,ou=Development,dc=acme,dc=org))
can be used to find the appropriate entry. Please note that the LDAP server has to support this recursive membership searches, a feature that usually has to be configured separately as special module or overlay of the LDAP server.

3.19.1 Transparent Failover with Redundant LDAP Servers

When your infrastructure provides multiple redundant and homogeneously defined LDAP servers, the password authorization can make use of them to ensure transparent failover. Homogeneously defined means that the servers differ only in terms of hostname or ip-address. All other connection parameters and properties must be identical on all of the servers. To configure such a group of servers, enter their hostnames or addresses as a comma-seaparated list in the **LDAP Host** field, e.g.:

- LDAP Host: ldap1.acme.org,ldap2.acme.org,ldap3.acme.org

All the other properties are to be entered in the same manner as for the single server case described above.
When the system starts, it marks the first server of the list as the current default one. All password verification attempts will use this server. If the server is not available, the next server in the list is checked (after a timeout, c.f. property **Timeout (ms)**). During one login attempt, there will be at most one connection attempt to each of the LDAP servers. Being *N* the number of entered LDAP hosts, the duration of a login attempt may be *N*Timeout* milliseconds if all your LDAP servers are down.
If the connection to a server has failed, another server is designated as the current default server. During "normal" operation, the last server that has been designated as the default one will be used again and again, until a connection attempt to it fails. This "sticky" behavior ensures that a server recently known to be operational is being tried first with a good probability to succeed thereby without any timeout penalties to pay in the normal case.

3.20 Change administrator password

With this link you can change the password of the *sysadm* user. The corresponding parameter in *avw.conf* is *avw.syspwd*. The default password is *digital* (after a default installation of **@enterprise**).

4 Patching and Upgrading your Installation

This chapter describes the patching and upgrading mechanism of **@enterprise**. Some terminology first:

- **Version:** A version is the number of the **@enterprise** version, e.g. *10.0*.
- **Build:** Represents a combination of a version and a revision number, e.g. *10.0.6778*. The revision number can be found e.g. in the **@enterprise** changelog.
- **Patch:** This term is used, if the build number has been or is to be increased, e.g. updating from **@enterprise** 10.0.6770 to 10.0.6778.
- **Upgrade:** This term is used, if the version of **@enterprise** has been increased, e.g. upgrade from **@enterprise** 9.0 to 10.0.

4.1 Patching the Installation

To assure the quality and reliability of your installation, bug fixes and enhancements for **@enterprise** are provided in the form of patches. The main starting point for obtaining such patch files is our download area reachable via **http://www.groiss.com/download.html**.
Occasionally for special circumstances, we might provide new versions of single artefacts of a revision, but the main distribution format for patches are patch archives. The technical format of a patch archive is a ZIP-archive.
Patch archives are named like *patch-{version}.{revision}.zip*, e.g. *patch-10.0.6778.zip*.
A patch archive is a bundle which incorporates the individual new versions of the files to be patched, along with two additional files (*version*,*changes*) which describe the patch and the needed actions.
The *version* file contains a *base* revision number and a *new* revision number. A patch archive is applicable to an installation (of the same version), if the current revision number of the installation is between the *base* revision of the patch archive and the *new* revision. If the system is not at the minimum required build, or the revision of the patch is not higher, or if the current build of the installation cannot be determined, then the patch is not applicable.
The *changes* file of a patch archive describes the actions on the file system that are needed to bring up the installation up to the new revision (copying of new files and deletion of unneeded files).

There are two alternative ways to incorporate the patches into your installation. We will first describe the automatic procedure and then a more manual way.

4.1.1 Automatic Patch Method

The automatic patch method requires the administrator to place the patch archive into a a special location (the ./patches folder), to stop @enterprise, to initiate the upgrade by starting @enterprise in the upgrade mode[1], and to restart @enterprise normally.
What's done automatically is the check of the applicability of the patch archive to the current installation according to the revision number as stated above, and to execute the appropriate operations on the file system according to the *changes* file in the patch archive.
The file operations are playing it safe, by preventing the loss of files which are in your current installation. Before a patch archive is applied, a backup of all affected (deleted /changed) files will be performed to the backup folder (*./patches/backup/{timestamp}*). For each application of a patch, a separate backup folder will be created, it will not be deleted by @enterprise.
The procedure is as follows:

1. Download the patch archive from our web site.
2. Its wise to play it really safe, so we strongly suggest a backup of the installation and the database.
3. Copy the patch archive file to the ./patches folder of the installation. It should be the only patch file there.
4. Optionally you can check the effects of the patch archive prior to applying the patch. The system administration provides a function in the System-Control section. This is especially handy to identify files that have been overwritten in the local installation. If there are any clashes, you will have to make sure that your local changes are not lost by manually reapplying them after the patch action or by updating the local files according to the changes in the original base files.
5. The patch procedure can then be initiated by:
   - If you are using **@enterprise** in standalone mode (Jetty), you have to start the server in upgrade mode,by stopping @enterprise followed by calling the corresponding start script (*ep.bat* or *ep.sh*) with the single *-upgrade* argument. **Please note:** when running as service (either in Windows or in Linux), stop the service first.
   - If your installation runs in an application-server (e.g. Apache Tomcat) use the provided patch-script (upgrade.bat/upgrade.sh) to apply the patch.[2]
     Please note: for reasons of file locking, your application-server, or at least the **@enterprise** application must not be running while applying the patch.
     Please note: to use your desired java version during the patch, the path to the

[1] For historical reasons this is named *upgrade*, a more consistent naming would be *patch* according to the terminology at the beginning of the section

[2] Those scripts just apply the patch at the file system and do not really start @enterprise. After completion of the scripts, you can start @enterprise in your usual way.

directory containing the java executable can either be prepended to the call like stated below or the scripts can be changed accordingly. If no such path is provided, some system default java version will be used.

- **Windows**: set "JAVAPATH=c:\jdk\x.y\bin" && .\upgrade.bat
- **Linux**: JAVAPATH=/usr/opt/java/x.y/bin ./upgrade.sh

6. This action starts the replacement of the files and also applies needed changes to the database.
7. After the patch has been successfully applied, the patch archive is moved to the corresponding backup folder and all actions are logged in the *./patches/patch.log* file.
8. Start @enterprise in normal mode by your usual procedure.

4.1.2 Manual Patch Method

The manual patch method differs from the automated one with respect to operations on the file system. All changes have to be done manually (or by some other means out of the scope of @enterprise). New files must be copied to to their appropriate locations, old or obsolete files must be removed from the installation. The *changes* file of a patch archive can be used a a guiding line here.
The following steps are to be executed:

1. Stop the server.
2. Its wise to play it really safe, so we strongly suggest a backup of the installation and the database.
3. Manually apply the operations on the file system / installation directory.
4. Start the server with the *-upgrade* option. Any required database steps will be performed.
5. Start the server again without the *-upgrade* option.

4.2 *Upgrading/Patching an* @**enterprise** *Application*

Applications in @**enterprise** can be kept up to date using the same mechanisms as for the base system itself.
The upgrade/patch of an application contains of a set of files which must be replaced in an installation. @**enterprise** also offers the possibility to execute further actions via an upgrade method. Typical actions include:

- XML import: Master data of the application can be adapted.
- Execution of database scripts.
- Other JAVA methods.

The *upgrade* method is part of the application class which has to implement the interface *com.groiss.wf.ApplicationAdapter*. The application will be upgraded/patched automatically, if you start your server with disabled logins or the *-upgrade* option. Further information can be found in the API of **@enterprise** (ApplicationAdapter.getVersion() and ApplicationAdapter.upgrade()).
The steps for performing a manual application upgrade/patch are the same as manually installing a patch for the base system.
Patches for applications can also make use of the automatic patch mechanism. Place your patch-file in the patches folder in your application directory. The filename must match *patch*.zip*. When starting the server with disabled login or the *-upgrade* option, these patches will be applied too. Please note that it's not recommended to patch more than one application at a time. For further information on how to build your own patch-archives see the programmers manual.

Hint: It depends on the application which upgrade procedure is used! In some applications it is necessary to execute the *upgrade* method only, other applications need to be upgraded manually (steps should be described in an own "readme" file).

4.3 Performing an Upgrade of @enterprise

This section describes the steps needed to upgrade from a prior **@enterprise** version to the current one (e.g. 9.0 to 10.0):

1. Backup your old installation and database.
2. Extract the content of setup100.jar into a new directory. We recommend to use the initial setup wizard for this purpose.
3. Copy your existing configuration file (avw.conf or avwservlet.conf), the forms directory, required jar-files (e.g. JDBC driver) of the *lib*-directory (e.g. ojdbc7.jar) to the corresponding directories of the new version.
4. To perform an interactive upgrade, start the server and login as sysadm. You should now be redirected to the upgrade page where you can initiate the necessary upgrade procedure for the database.
5. To perform an unattended upgrade, start the server with the *-upgrade* option. Any required database upgrades will be performed. The server will be stopped automatically.[3]
6. (Re-)Start the server and delete Browser caches.

Hint: If problems with your forms occur after upgrading to current **@enterprise** version, try to set parameter *ep.xforms.dojo.load.sync* in section *Other parameters* to value true.

[3] If you are using PostgreSQL as your DBMS, it might be necessary to use this upgrade variant and refrain from using the interactive one.

4.4 Application repository

@enterprise offers the possibility to get new applications and patches from an application repository (the user interface is described in the *System Administration Guide*). For this purpose an application for managing these files is necessary - an example for a repository management application can be downloaded via http://www.groiss.com

4.4.1 Interface

The application repository defines an URL (can be entered under *Communication/Application Repository URLs*). A HTTP request on this URL returns a XML that accords to defined schema *applrepository.xsd* (can be found within *ep.jar*).

Example for XML:

```
<applications>
 <application>
  <id>pm</id>
  <name>Project Management</name>
 <icon>com.groiss.apprep.AppRepository.showResource/10.0/pm/pm.png</icon>
  <shortdesc>Project Management ...</shortdesc>
  <description></description>
  <version>1.0.1.61</version>
  <changelog></changelog>
  <patchfile>
   com.groiss.apprep.AppRepository.showResource/10.0/pm/patch-1.0.1.61.zip
  </patchfile>
  <url>com.groiss.apprep.AppRepository.showResource/10.0/pm/pm.zip</url>
  <timestamp>2012-10-16T14:34:32Z</timestamp>
 </application>
...
</applications>
```

The XML defines some parameters of the application (<*id*>, <*name*>, <*shortdesc*>, <*description*>, <*version*>) and links to an icon (<*icon*>), a download- (<*url*>) and a patch-file (<*patchfile*>). It is also possible to define a changelog file which contains information about the changes in current patch.

4.5 Migration of deprecated DBMS data types

4.5.1 Migration of Oracle data type LONG

This section describes the migration steps for existing Oracle 9i and newer installations, because since Oracle 9i LONG is deprecated. Existing installations can continue to use the old data types (not recommened), but new installations should use the new data types. Following steps must be performed for migration:

Hint: Please note that the migration is at your own risk and can take a long time!

1. Backup your old database!
2. Write down the indices of the affected tables. Following query helps to get a list of the affected indices wrapped in executeable *alter index* statements:

```
select 'alter index '||index_name||' rebuild;' as command
from user_indexes where table_name in (
  select table_name
  from user_tab_columns
  where (table_name like 'AVW%' or table_name like 'FORM%')
  and data_type  in ('LONG', 'LONG RAW')
)
order by index_name;
```

   Do not execute the *alter index* statements in this step!

3. For each table which contains one of the previous mentioned data types an *alter table* statement must be performed like in following way:

   - alter table <tn> modify (<longcolname> clob default empty_clob());
   - alter table <tn> modify (<longrawcolname> blob default empty_blob());

   It is a little bit cumbersome to get the affected tables. For this purpose, the following query helps to get a list of affected tables wrapped in executable *alter table* statements as mentioned above:

```
select
   'alter table '||table_name||
   ' modify ('||column_name||' '||
   (case when data_type='LONG' then 'CLOB' when data_type = 'LONG RAW'
     then 'BLOB' else 'ERROR' end)|| ' default empty_'||
   (case when data_type='LONG' then 'CLOB' when data_type = 'LONG RAW'
     then 'BLOB' else 'ERROR' end)||'());'
    as command
    from user_tab_columns
where (table_name like 'AVW%' or table_name like 'FORM%')
and data_type  in ('LONG', 'LONG RAW')
order by table_name, column_name;
```

4. Perform the *alter index* statements which have been created in step 2) to rebuild the indices.

4.5.2 Migration of deprecated MS SQL-Server data types

Since MS SQL-Server 2005 Microsoft has been set some data types to deprecated and replaced them by new ones. The following list contains the deprecated and the appropriate new data types:

- text has been replaced by varchar(max)
- ntext has been replaced by nvarchar(max)
- image has been replaced by varbinary(max)

Existing installations can continue to use the old data types, but new installations should use the new data types. This section describes the recommended migration steps for existing installations with MS SQL-Server 2005 and newer:

Hint: Please note that the migration is at your own risk and can take a long time!

1. Backup your old database!
2. For each table which contains one of the previous mentioned data types an *alter table* statement must be performed like in following way:
   - alter table <tn> alter column <textcolname> varchar(max);
   - alter table <tn> alter column <ntextcolname> nvarchar(max);
   - alter table <tn> alter column <imagecolname> varbinary(max);

   It is a little bit cumbersome to get the affected tables. For this purpose, the following query helps to get a list of affected tables wrapped in executable *alter table* statements as mentioned above:

```
select
'alter table '+table_name+
' alter column '+column_name+' '+
(case when data_type='text' then 'varchar(max)'
      when data_type='ntext' then 'nvarchar(max)'
      when data_type = 'image' then 'varbinary(max)' else 'ERROR' end)+
';'
as command
from information_schema.columns
where (table_name like 'AVW%' or table_name like 'FORM%')
and data_type  in ('text','ntext', 'image')
order by table_name, column_name;
```

Heterogeneous data types for %OIDTYPE% patterns

Installations using MS SQL-Server as database engine might suffer from heterogeneous types being used for %OIDTYPE% patterns.

The old DB translator (used for SQLServer version < 2000) uses "DECIMAL(20)", the new one (for SQLServer versions >= 2005) uses "BIGINT". In really ancient installations, the type "DECIMAL(28)" might also be in use.

As a consequence, there might be installations with more than one type being used for oid columns or columns referencing them. This has not been a problem up to recently.

But in **@enterprise** 9.0 we began to use referential integrity constraints (foreign keys) in our schema and were hurt by a peculiar behavior of SQL-Server which insists that the data types of referencing columns and referenced columns must be exactly the same.

So, depending on the configuration and history of the installations there might be problems when upgrading to a recent **@enterprise** version because of nonuniform data type usage.

To remedy this potential problem, two measures have been implemented:

1. The new translator has been extended to determine the oidtype at startup. Essentially, if the oid columns of all avw* and form* tables are using the same data type, then this data type is being used as %OIDTYPE%. If not, BIGINT will be used.

   With this change, all installations which have not yet switched to the new translator might do so without being affected by the issue.

2. For installations which already do use different data types for %OIDTYPE%, we provide a powershell script which generates the needed SQL DDL for type migration to BIGINT.

   Please contact support@groiss.com for further information regarding the script.

5 Clustered @enterprise System

5.1 Overview and Principles of the Clustered Architecture

The clustered architecture supersedes the previous distributed architecture. The aim of the new architecture is to allow for

- increased scalability,
- increased availability,
- easier configuration,
- more flexible operation.

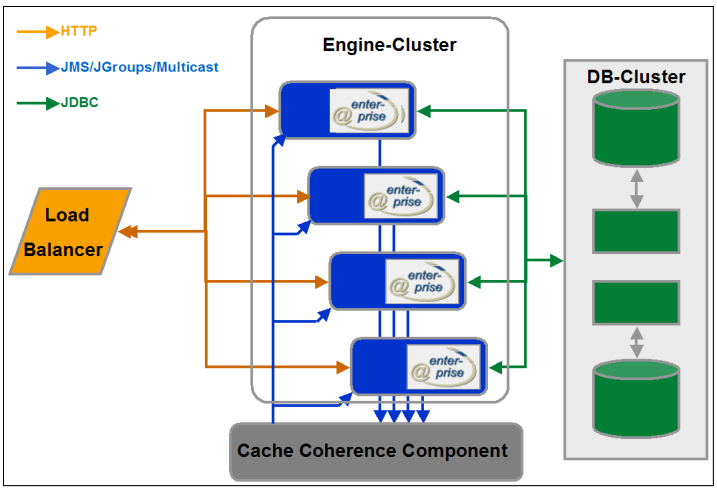

Figure 5.1: **Cluster Architecture**

Figure 5.1 shows the principal layout of such a cluster. The logical architecture consists of a set of **@enterprise** engines (termed "nodes") which access a common database and are operated in a peer to peer mode to a large extend. A load balancing mechanism is employed to ensure even load distribution within the cluster. Consistency between the caches in the nodes is ensured by a cache coherence service.

While there are no single points of failure within the cluster nodes, we require the database to be available and scalable to an extend that imposes no bottlenecks for the rest of the system.

5.2 Cluster and Nodes

As already mentioned, a node is a single Java Virtual Machine instance. In a typical production environment, there will be one node running on a single physical machine. In a development or test environment, more than one node could be running on one machine (without enhanced scalability and availability).
The cluster is represented by a single entry in the Server section of the administration. Each node is identified by a Node-Id which must of course be unique within the cluster. Nodes can enter and leave the cluster at runtime. New nodes can be added to the cluster on the fly.

5.3 Configuring a clustered @enterprise System

The clustering of an **@enterprise** system will typically comprise of the following actions

- configuration of the underlying platforms in terms of hardware, operating system, network and database connectivity and JVM,
- installation of a single (nonclustered) **@enterprise** system,
- selection of the appropriate transport mechanism for the cache coherence service, its configuration and startup if necessary,
- distribution of the **@enterprise** installation directory to the nodes,
- adapting the **@enterprise** configuration (optionally using configuration in database),
- starting the nodes.

Details for each of the steps can be found in the following sections.

5.3.1 Platform Configuration

The nodes of an **@enterprise** cluster can run on a heterogeneous platform as far as the hardware and operating system is concerned. While it is also possible to use different versions of the JVM/JDK it is strongly recommended to use the same principal version for each node. If your installation must use different versions, intense testing is strongly advisable.
The requirements for the minimal technical layout of the nodes do not differ from the layout of a single machine. A possible exception are the network interface requirements. It may be advisable to use different physical network interfaces and interconnections for client connections, database connections and possibly for the cache coherence service.

5.3.2 Installation of a nonclustered System

No special issues are arising here because of the cluster. Just install a plain **@enterprise** system and make sure that it is working.

5.3.3 Adapting the @enterprise Configuration

Configuration: Under Configuration / Classes / Services, an entry for the cache coherence service must be added as the last service:
com.groiss.dbcache.coherence.CoherenceService cs

The following configuration entries are needed in a clustered node under Configuration / Cluster or *avw.conf*:

- **Clustering enabled - avw.cluster.activated:** Must be checked.
- **Server name - avw.servername:** Name of the server. Must be the same on each node of one cluster.
- **Node Id - avw.node.id:** Id of the cluster node. Must be unique within the cluster.
- **Performance factor - avw.node.perffactor:** Relative performance factor of the node. Depends largely on CPU power of the node. A node with a factor of 2 is expected to support twice the users of a node with factor 1. The load balancer makes use of the factor to distribute user sessions according to the relative power of the nodes.
- **Member of load balancing - avw.node.loadbalancing.member:** If set to *YES* (by default), the loadbalancing function for this node is active, i.e. the node is a potential target for clients which request loadbalanced sessions. On nodes which serve special purposes and should not receive logins from "ordinary" clients, this parameter should be unchecked.
- **Set Node Cookie - avw.cluster.setnodecookie:** Needed for load balancing in proxy environment (see chapter 6).
- **Coherence strategy - avw.dbcache.coherence.strategy:** Currently there is just one strategy supported: Notification. Do not confuse this with the client notification mechanism. While the things share the same name, they have nothing in common. In the future, other strategies might be provided as well.
- **Transport layer for Coherence - avw.dbcache.coherence.transport:** Choose the appropriate transport mechanism like described above.
- **Disallow logins after coherence error - avw.cluster.coherence.error.disallowslogin:** If this checkbox is activated, no logins are possible anymore in case of a coherence error.

Hint: In section *Other parameters* the configuration parameter *ep.timerentry.change.check.seconds* can be used to check for changes in timer entries.

Ports: If you do run several nodes on one machine (e.g. for testing purposes), ensure that distinct network port numbers for the HTTP server, the HTTPS server and the RMI-mechanism are used.

Directories: If your nodes run on the same machine or access the same remote file systems, be sure to configure each of the nodes with distinct destinations for the log file and the error log file as well as a distinct temporary directory.

Timers: Timers require special consideration in a cluster. There might be timers which should run on each node, and there might be timers that should only be running on one dedicated node of the cluster. The former timers must just be marked by checking the box *Run on each Node* on the timer edit form.
The latter ones must be marked by NOT checking the box and require special action. In a clustered system one of the nodes assumes responsibility for running the timers. Transparent failover is provided.
To enable this functionality, make sure that two timers are started on each node:

- **HeartBeat:** Should be running on each of the nodes. Periodically writes a timestamp to the database. Used to monitor cluster nodes. During normal operation, there is exactly one update of a single row followed by a commit per heartbeat (and node). The heartbeat mechanism uses a dedicated database-connection when more than five database connections have been configured for the node eliminating hold-ups from finding a connection and overhead from frequently releasing and reacquiring the connection. Recommended periods are in the range of 3 to 10 seconds. Because of these short heartbeat intervals it is recommended to use a dedicated timer thread by assigning a unique thread-id (e.g. "heartbeat") to the timer. This avoids the possible delay of the heartbeat by other (longer-running) timers, thereby getting the heartbeat info to the database as fast as possible.

- **ClusterCheck:** Should be running on each of the nodes. Periodically checks health state of the cluster. Recommended periods are in the range of 120 to 600 seconds. There are two aspects to check for. First, if a node fails to update its timestamp within the tolerance time defined in the *Clustercheck Tolerance* parameter, its state is set to not running. Second, if none of the nodes runs the timers which are started just once for the whole cluster, one node must assume this role.

5.3.4 Optional synchronization of configuration via the database

The configuration files of **@enterprise** (e.g. avw.conf) and of the applications (appl.prop) can be mirrored in the database to facilitate cluster-wide synchronization of common parameters in those files. For using this functionality in a cluster, the following configuration steps should be performed (under the assumption, that a clustered system is already deployed and working):

1. Create an additional configuration file for each cluster node - e.g. *node_N.conf* (N is the Node Id defined in configuration parameter *avw.node.id*) - which contains the node specific configuration parameters that should **not** be synchronized, especially the configuration parameter *avw.node.id* must be entered in each node specific file!

All other configuration parameters which should be synchronized between the cluster nodes must be stored in the cluster wide *avw.conf* or *avwservlet.conf*. Each parameter should either be placed in all node-specific configuration files or exclusively in the cluster wide configuration file. Each node-specific file must have a unique name and should be placed just on the corresponding node.

Example configuration of *node_N.conf* in a clustered environment where the nodes are on different machines in the network (typically in a production environment):

```
#essential
avw.node.id
#optionally
avw.node.perffactor
avw.node.loadbalancing.member
```

Example configuration of *node_N.conf* in a clustered environment where the nodes are on the same machine (e.g. test environment):

```
#essential
avw.node.id
#optionally
avw.node.perffactor
avw.node.loadbalancing.member
avw.formclassdir
Httpd.tempDir
logger.logfile
logger.errorfile
#if Jetty/standalone deployment is used
httpd.port
http.ip-address
ssl.port
ssl.ip-address
httpd.admin.port
httpd.admin.ip-address
```

2. Define a comma separated sequence of paths to the appropriate configuration files depending on the deployment scenario

   - Standalone deployment with internal Jetty server: in *ep.bat* resp. in *ep.sh* or - if using a Windows service - in *wrapper.conf*
   - Deployment within an Application server: in *WEB-INF/web.xml* of **@enterprise**.

   Hint: The file *avw.conf* or *avwservlet.conf* must be always the last file in the sequence. In this file all new parameters will be inserted when the configuration is changed (that is, new parameters will be in the cluster-wide scope).

   Example configuration in ep.bat/ep.sh:

```
"%EP_JAVACMD%" -Xms16m -Xmx256m -Djava.awt.headless=true
 com.groiss.component.Bootstrap conf/node_cn1.conf,conf/avw.conf %1
```

Example configuration in wrapper.conf (located in folder *service*):

```
wrapper.app.parameter.2=conf/node_cn1.conf,conf/avw.conf
```

Example configuration in web.xml:

```
<servlet>
 <servlet-name>AVWInit</servlet-name>
 <servlet-class>com.groiss.servlet.AVWInit</servlet-class>
 <init-param>
  <param-name>conffile</param-name>
  <param-value>conf/node_cn1.conf,conf/avwservlet.conf</param-value>
 </init-param>
 <init-param>
  <param-name>standalone</param-name>
  <param-value>false</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
```

3. Set parameter ep.configuration.store.in.database=true in *avw.conf* or *avwservlet.conf*. This parameter is also available in section *Configuration/Other parameters*.

If the configuration steps have been performed successfully for each cluster node, the cluster is ready for use now. During node startup, the following steps are performed automatically:

- Read configuration files first
- Start the DBConnPool to get access to database
- Synchronize parameters between database and configuration files by means of the file date (maximum modified time and change date from header) and attribute "changedat" of the database table *avw_config*
- Reload configuration and continue startup

If a configuration parameter is changed via **@enterprise** administration GUI, the changes are written in the corresponding configuration file in the file system at the current cluster node and also in the database. If a configuration parameter is changed directly in a configuration file when a cluster node is running, the function *Reload configurations* in **@enterprise** administration under *Admin-Tasks/Server/Server control* must be performed to synchronize the configuration files and the database for this cluster node.

No automatic synchronization of configurations between cluster nodes is intended. All synchronizations must be triggered manually via the function *Reload configurations* on mask "Server control" (see *System Administration Guide* of **@enterprise**), i.e. if configuration

changes (with intended cluster-wide scope) have been made on cluster node 1, then on all other cluster nodes, the function *Reload configurations* must be performed manually to synchronize the configuration for those nodes.

Please note that changes to parameters needed by the synchronization mechanism itself (the parameter *ep.configuration.store.in.database* and the JDBC parameters (database.driver.class, database.url, etc.)) might require additional manual synchronization steps on each cluster node.

Please do also keep in mind, that no conflict handling of the cluster wide configurations in the database is implemented. The last writer of such a configuration will win. Let us give an example: if the value of parameter "x" has been changed to "v1" on cluster node 1 and is subsequently and independently changed on cluster node 2 to "v2", the value "v2" for "x" will finally be stored in the database. If later on function *Reload configurations* is performed on cluster node 1, value "v2" will be stored for parameter "x" in the configuration file of cluster node 1 (instead of originally intended value "v1"). It is recommended practice to routinely check if the configuration is up to date (by means of the mask "Server control" mask) before changing it.

As already mentioned, the configuration files for applications (*appl.prop* files) are subject to the same synchronization mechanism. All parameters in those files are considered to be of cluster wide scope.

5.3.5 Transport Mechanisms for Cache Coherence Service

The cache coherence mechanisms task is to propagate cache relevant events within the cluster in order to keep the caches current. For the time being, the following event types are propagated:

- **Workitems:** Changes in the worklist (new items, finished items, ...)
- **Substitution:** Changes in substitutions of users (new substitute, period of substitution starts or ends)
- **Seen Objects:** Items that are new to a user.

We provide the following choice of transport mechanisms to account for different needs of an installation:

- Unreliable Multicast via UDP
- Reliable Multicast via JGroups
- Java Message Service (JMS)

Unreliable Multicast via UDP

While this mechanism is easy to configure and poses virtually no overhead, it is recommended primarily just for development or test installations, due to possible loss of packets. A installation which uses dedicated physical network interfaces and interconnections

for cache coherence service might also use unreliable multicast with good results, but one should be aware of the susceptibility to errors. This transport mechanism uses features present in the Java platform, no deployment or startup is needed.
The following configuration parameters are needed under *Configuration → Coherence* and must be identical on all cluster nodes. These parameters are available only, if *Standard Multicast* is used as *Transportlayer for Coherence*:

- **Multicast-IP-Address:** Must be a valid multicast address. No two clusters should use the same multicast address. Be aware of other applications using multicast in your configuration. For specification and assignments of multicast addresses, refer to http://www.isi.edu/in-notes/iana/assignments/multicast-addresses. Monitoring of multicast packets is quite easy with tcpdump ("tcpdump ip multicast").
- **Multicast IP Port:** Port to send and receive multicast packets. Must be available on the machine.
- **Multicast TTL:** Determines the scope of multicast packets on the network. For clustered systems with small "network diameter" this should be 1.
- **Buffersize (Bytes):** Size of reception buffer in bytes. Recommended value is at least 30000 Bytes.

When specifying these values, be aware of possible address space collisions with a multicast based client notification service or cache coherence services of other clusters.

Reliable Multicast via JGroups

JGroups is an open source communications library for reliable group communication. It is written in Java (www.jgroups.org). It is deployed in the **@enterprise** engine itself and needs no external processes running. It is started automatically. The library itself consists of a single Java archive named jgroups-all.jar and uses the apache commons logging facility (commons-logging.jar), which must be explicitly added to the classpath (mere placement in the lib directory is not sufficient).
The following configuration parameters are needed under *Configuration → Coherence* and must be identical on all cluster nodes. These parameters are available only, if *JGroups* is used as *Transportlayer for Coherence*:

- **Groupname:** JGroups has the notion of communication groups. A member must state the groups he belongs to. Can be an arbitrary string, we recommend to use *epgroup* or to use the name of the server entry in the cluster.
- **Properties:** This parameter specifies the location of a configuration file in XML-syntax.

The recommended configuration is located in the jgroups/ccs.xml file. Since the whole JGroups protocol stack is configured through it, it looks rather complicated. But in normal situations, just a handful of key parameters need to be changed. Such parameters are clearly marked in the ccs.xml file. The parts of the configuration to be changed are the multicast IP address mcast_addr, the multicast port number mcast_port, and the time to live ip_ttl.

For the multicast address and multicast port we refer to the previous section about unreliable multicast, for the time to live we recommend either 1 as the packets should only reach the other **@enterprise** node which are placed in the network vicinity. If network components are between the nodes of the cluster, it might be necessary to increase this value to 32. In case of doubt, consult your local network administrator. Please avoid any interference within **@enterprise** (e.g. client notification service with multicast) when selecting multicast parameters.
The other properties in the file should not be changed without intimate knowledge about JGroups.

Java Message Service (JMS)

The usage of JMS for the transport of cache coherence messages can be characterized as follows. The publish subscribe paradigm is used. Per node there is one subscriber and one publisher. All nodes subscribe to the same topic. No message selectors are used. We use non-persistent, auto-acknowledged, non-transacted messages and nondurable subscribers.
JMS does not run within an **@enterprise** JVM, it must be configured and started separately. Apache ActiveMQ (http://activemq.apache.org) meets all requirements and is known to be reliable, but virtually any JMS implementation should be suitable. Hints for configuring a particular JMS may be obtained via the **@enterprise** support.
The following configuration parameters are needed under *Configuration* → *Coherence* and must be identical on all cluster nodes. These parameters are available only, if *JMS* is used as *Transportlayer for Coherence*:

- **JMS Provider URL:** The URL name of the JMS provider. For ActiveMQ this is something like tcp://<jmshost>:61616?wireFormat.maxInactivityDuration=10000.
- **JMS ContextFactory:** Name of the Java class for construction of the JNDI-Context. For ActiveMQ this is org.apache.activemq.jndi.ActiveMQInitialContextFactory.
- **JMS TopicConnectionFactory:** Java class name for the topic factory of the JMS provider. For ActiveMQ this is ConnectionFactory.
- **JMS Topic:** The name of the topic used for communication. Such topics must typically be created within an JMS provider by the administrator. For ActiveMQ this can also be a dynamic topic like dynamicTopics/avw.
- **JMS Time to Live (ms):** The JMS provider is free to throw away messages which are older than this timespan. Should be in the range of 30 to 120 seconds. Unsynchronized clocks on participating systems might inhibit proper message transfer.
- **JMS Username:** Name of the user which is utilized for communication with the JMS provider. If this parameter is left empty, an anonymous connection is established. User administration is specific for each JMS provider.
- **JMS Password:** Password for the user mentioned before.

More than one cluster can use a JMS provider, if the names of the topics are kept unique for each cluster. Do not use the same topic name for client notification via JMS and for client notification if you are using the same physical provider for both purposes.

5.4 Operation of a clustered system

5.4.1 Monitoring

A cluster health monitor which displays the state for each of the nodes can be accessed via Admin-Tasks / Server / Running Nodes Monitor.
The fields displayed are:

- **Hostname:** Name of the cluster.
- **Node-Id:** Id of the node.
- **Start Time:** Time of startup of this node.
- **Last HeartBeat:** Timestamp of last heartbeat made by this node.
- **Running:** Marks, if the node is running.
- **ClusterTimers:** Marks, if the node is the one which runs the cluster timers.
- **Load:** Current number of connected users.
- **Performance Factor:** The performance factor of the node.
- **Load Coefficient:** The current load coefficient (number of users divided by performance factor).
- **Load Balanced:** Marks, if the node is member of loadbalancing (see section 5.4.2).
- **Logins enabled:** Marks, if new logins are allowed on the current node. Logins can be enabled/disabled with the toolbar-function *Disable/Enable Login*.
- **Current Session:** Shows, if current sessions should be kept, renewed or aborted. At startup this is always set to "keep".
- **Successor Nodes:** The id of the successor node is displayed where the clients of the "switched off" node should be logged in, if login is restricted (see column *Logins enabled*) and current session should not be kept. At server startup this column is always empty.

Hint: *Current Session* and *Successor Nodes* are used by **@enterprise** Java Clients only!

5.4.2 Load Balancing

Principle A client which wants to obtain a load balanced session should first connect to a special URL on an arbitrary running cluster node. There, the client will be redirected to the least loaded node (HTTP(S)-Client) or can obtain appropriate initial URLs of this node (RMI-Client). Please note that this mechanism does not necessarily imply the use of a dedicated front-end load balancing system. Details for such a configuration can be found in chapter 6.

HTTP(S)-Clients The URL for getting a load balanced session for an HTTP(S) Client is:
http[s]://<host>:<port>/<context-root>/
servlet.method/com.groiss.avw.html.HTMLNodes.redirect

The client will be redirected immediately to the server with the lightest load.

RMI-Clients Use the same mechanism as mentioned above. A client should open an URL-Connection to:
http://<host>:<port>/<context-root>/
servlet.method/com.groiss.avw.html.HTMLNodes.redirectJavaClient

Three URLS are returned, each in a separate line. The URLs can be used by the client to obtain an appropriate session to the node. The first URL is the one for HTTP clients, the second one is the URL for RMI clients, the third one is the URL for the HTTPS clients. This data can be used in the client to obtain a session to that node.

5.4.3 Event Handling

Event handlers are executed on the node where the event has been raised.

6 @enterprise *in a Load balancing / Reverse proxy environment*

We will briefly discuss some key aspects of deploying an **@enterprise** clustered system behind a load balancing / reverse proxy.

6.1 Basic constellation

A cluster consists of several **@enterprise** nodes; each with its own unique node id. Each node provides HTTP(S) services to users addressed via a host name / port combination respectively an ip-address / port combination. All nodes together comprise the cluster (which is - somewhat confusingly - for historical reasons and called the "Server").
Installing a reverse proxy between the nodes and the clients strives to implement three main functions:

- Providing a node transparent view of the **@enterprise** system
- Load balancing
- SSL termination

6.2 Main technical considerations

6.2.1 HTTP session binding (sticky sessions)

In an **@enterprise** cluster, requests for a client in the scope of a session need to be bound to an arbitrary, but particular node. So **@enterprise** needs a proxy that supports session binding (also called sticky sessions or persistent sessions).

A session in **@enterprise** is conveyed in the form of a session cookie which name is <serverid>_EPSESSIONID. In order to provide session binding to nodes, a second cookie is issued. This routing cookie is named <serverid>_EPNODEID, its value is the node id of the node the session is bound to. It is set at login time and deleted at logout time. The proxy must be configured to honor the node id cookie inasmuch that incoming requests providing the cookie must be routed to the corresponding node.

6.2.2 HTTP session failover

@enterprise provides no out of the box means for session failover. When a node fails, the proxy has to detect this state and will reroute the request to use another (working) node. Since the HTTP session is not known on the new target node, the user will have to login again to this node.

6.2.3 Node election at initial session creation

At the first contact between client and the **@enterprise** cluster, the cookie bearing the node id has not been set. So the choice of the initial node is somewhat arbitrary. The proxy should be configured to use some sensible strategy. By using a special login URL (see later), the **@enterprise** load balancing mechanism, which is based on a combination of node weight and user session count, can also be used for initial node election.

6.2.4 SSL termination in Proxy

In a configuration where all traffic between the proxy and the clients is carried out via SSL/TLS, a function of the proxy is to terminate the SSL traffic and to use plain HTTP to connect to the cluster nodes. This diminishes the SSL processing load at the nodes.

Since this would be within the perimeter of a central network, unencrypted data traffic in this network links should not be a security issue. Should SSL termination at the proxy really not be allowed, a different configuration is needed, since the cookies are also encrypted in such a scenario and are therefore not accessible for the proxy for node routing purposes.

6.2.5 Transparent view for the clients

To present a uniform an node transparent address for the clients, the **@enterprise** server object must be configured accordingly. The "official" address of the clustered system must be set. In particular, the combination of protocol (HTTP or HTTPS), host name and port (which will be used by the client to contact the proxy) must be provided via *Administration/Admin-Tasks/Cluster/Servers*.

If the nodes are configured to use dedicated admin connectors, the proxy configuration as well as the server info must be augmented accordingly.

6.2.6 HTTP header transformation by the Proxy

In order to provide the nodes with appropriate data about the technical nature of each request, the headers of the HTTP requests must be enriched by the proxy. Two additional headers are important: X-Forwarded-For and X-Forwarded-Proto. The first one bears the originating client address, the second one should be set to *https*, when the proxy uses SSL session termination, and the incoming connection was made via SSL.

6.2.7 Configuration considerations for @enterprise

The node routing cookie for session binding must be switched on via the parameter avw.cluster.setnodecookie in the *Cluster* section of the configuration.

An additional parameter httpd.jetty.behindproxy must be set in section *Other parameters* in the configuration; is accounts for correct interpretation of the X-Forwarded-* headers.

6.2.8 Special functions

Some special functions for a proxied configuration are provided via explicit URLs. The prefix for all those URLs is:

protocol://proxy:port/ctxroot/servlet.method/com.groiss.avw.html.HTMLNodes

Following functions are available:

- redirect: Use the **@enterprise** load balancing mechanism to contact the least stressed node (sets the node routing cookie)
- redirect?nodeid=<nodeid>: Explicitly set the node routing cookie to a dedicated node
- unbind: Remove the node routing cookie
- clientInfo: For troubleshooting purposes, all details about the request are presented

6.3 Example configuration

In the following, we outline the configuration of an **@enterprise** system with haproxy_1.8.X (available via http://www.haproxy.org/) as a reverse load balancing proxy.

The configuration should be seen just as a (working) starting point; there are literally dozens of proxy configuration options and tuning parameters which might need to be adjusted to derive a production ready configuration variant.

6.3.1 @enterprise constellation

We will use an **@enterprise** clustered system consisting of three nodes. Each node has two connectors: a HTTP user connector and an admin-connector which uses SSL (but is terminated by the proxy).

NodeId	Host name	User port	Admin port
Node_1	n1.acme.com	8001	8101
Node_2	n2.acme.com	8002	8102
Node_3	n3.acme.com	8003	8103

We assume, that the proxy will be reachable via:

Host name	User port	Admin port
enterprisewf.acme.com	80	443

So we have to set the **@enterprise** server object (c.f. subsection 6.2.5):

- *Protocol:* HTTP
- *Hostname:* enterprisewf.acme.com
- *HTTP(S) port:* 80
- *Administrative protocol:* HTTPS
- *Administrative IP port:* 443

We also need to set the parameters avw.cluster.setnodecookie and httpd.jetty.behindproxy as described above.

6.3.2 Preparation: Proxy building and SSL aspects

- *Compilation:* The options for the make command will have to be adjusted for the platform in use. For a reasonable modern Linux system, the following might be appropriate:

  ```
  make TARGET=linux2628 USE_PCRE=1 USE_OPENSSL=1 ADDLIB=-lz
  ```

- *SSL configuration:* The following steps can be used to provide the proxy with a self signed server certificate:

  ```
  # generate private key
  openssl genrsa -des3 -out privkey.pem 2048
  # generate certification request
  openssl req -new -key privkey.pem -out cert.csr
  # sign the certificate request
  openssl x509 -req -days 10000 -in cert.csr -signkey privkey.pem -out cacert.pem
  # the following two commands are needed, if haproxy startup should
  # not ask for manual input of the passphrase
  cp privkey.pem privkey_passphrase.pem
  openssl rsa -in privkey_passphrase.pem -out privkey.pem
  # build a certification chain
  cat cacert.pem privkey.pem > cacertprivkey.pem
  ```

6.3.3 Proxy configuration

In the following we provide a commented configuration file for *haproxy* for the installation outlined above on the host *enterprisewf.acme.com*:

```
# example configuration of haproxy 1.6.0
# as reverse loadbalancing proxy
# in front of a cluster

global
description cluster
```

```
# run in background
daemon
# user and group ids for execution environment
#user haproxy
#group haproxy
# empty unwritable jail dir for chroot
# chroot some_dir
# max number of connections per process
maxconn 1024
# logging definition syslog or systemd
# syslog
log 127.0.0.1 local1 notice
# systemd
# /dev/log local1 notice
# base directory for ssl stuff
crt-base /var/haproxy/cnf

defaults
# operating mode (layer 7 inspection)
mode http
# continuously update statistics; enable for testing; small performance impact
option contstats
# health checks are logged only when state of server changes;
# can be enabled in production, too
option log-health-checks
# timeouts; should be adjusted to match network and beackend configuration
timeout connect 5s
timeout client 30s
timeout server 30s
# use a long timeout for bidirectional tunnel traffic (e.g. websockets)
timeout tunnel 1h
# use default mode (changed in 1.6)
option http-keep-alive
# add x-forwarded-for header
option forwardfor
# use logging definitions from global section
log global

# listens on port 80 for incoming http requests (user connector)
frontend http-in
description user-port 80
bind *:80
default_backend ep-workers

# listens on port 443 for incoming https requests (admin-connector)
frontend https-in
description user-port ssl 443
```

```
# use all network interfaces ;
# the pem file is a concatenation of cacert and private key
bind *:443 ssl crt cacertprivkey.pem
# add  X-Forwarded-Proto header to mark request as secure for backend
reqadd X-Forwarded-Proto:\ https
# all requests use the following backend (the admin-connector)
default_backend ep-workers-admin


# the nodes (user-connectors)
backend ep-workers
description nodes (user connectors)
# use the node with the fewest connections
balance leastconn
# allow for redispatching a request; if a designated (via the cookie) server is down
option redispatch
# retries must be nonzero for redispatch to work
retries 3
# uses cookie <serverid>_EPNODEID for session affinity (sticky sessions)
cookie epcluster_EPNODEID nocache
# default options for all servers  of this backend
# check health via tcp ; weight for loadbalancing
# httpchk may be more adequate; see next section of this manual
# option httpchk ...
default-server inter 10s weight 10
# for each node a line must be inserted:
# might be better to use ip-Adresses instead of hostnames
# <Nodeid> <ip:user-port> cookie <Nodeid> check
server Node_1 n1.acme.com:8001 cookie Node_1 check
server Node_2 n2.acme.com:8002 cookie Node_2 check
server Node_3 n3.acme.com:8003 cookie Node_3 check

# the nodes (admin-connectors)
backend ep-workers-admin
description nodes (admin connectors)
balance leastconn
option redispatch
retries 3
cookie epcluster_EPNODEID nocache
default-server inter 10s weight 10
# <Nodeid> <ip:admin-port> cookie <Nodeid> check
server Node_1 n1.acme.com:8101 cookie Node_1 check
server Node_2 n2.acme.com:8102 cookie Node_2 check
server Node_3 n3.acme.com:8103 cookie Node_3 check

# admin connector for haproxy console
listen admin
```

```
description Administrative Overview
# user port 10001 on all interfaces
bind *:10001
stats enable
# default uri is:
# stats uri /haproxy?stats
stats uri /
stats realm HAProxy\ wfm \ Statistics
stats show-legends
# username:password
stats auth  admin:admin
stats admin if TRUE
```

6.3.4 Operation of haproxy

The *haproxy* server process can be started via haproxy -f <configfile>, but of course it should be integrated into the startup sequence of your server. An brief dashboard of the current state of the cluster according to the *haxproy* server can be obtained via
http://enterprisewf.acme.com:10001
As default, haproxy uses plain TCP health checks. As a consequence, a node appears to be available as soon as it accepts connections at the port of the HTTP connector. Since at this point in time, the node may still be starting up (e.g. loading the worklist cache), real operational readiness may be in effect somewhat later. Instead of the TCP check an HTTP check can be specified with the option httpchk in the haproxy config file. Put the following config options into each backend definition:

```
option httpchk GET /wf/servlet.method/com.groiss.cluster.ClusterInfo.getNodeStatusHAProxy
http-check disable-on-404
http-check send-state
```

Adjust the /wf prefix according to your context root. The lightweight getNodeStatusHAProxy method returns the following states:

- 200: refurned if node is loadbalanced and logins are enabled. Haproxy will mark this node as "UP/green".
- 404: if the node is loadbalanced, but logins are disabled and sessions should not be evicted. This means that haproxy will mark this node as "NOLB/blue", so it will disable the node for new connections but will continue to serve existing persistent connections to it.
- 403: this is returned when the node is not loadbalanced; either because it is configured in this way statically, or because loadbalancing is set to auto, which depends on state of worklist cache. Haproxy will mark this node as "DOWN/red".

7 Perimeter and Central Server

7.1 Rationale and Overview

7.1.1 Architectural considerations

Security considerations often lead to network separation in customer installations. There may be several forms of such a network architecture, but a common one is to separate the overall network into three logical zones:

- *Internal Zone:* this zone consists of all the internal assets and servers/clients. It is implicitly trusted and can be tightly controlled by the customer.
- *External Zone:* this is the outer world (the internet). No control, implicitly untrusted.
- *Demilitarized Zone (DMZ):* a zone between the external and the internal one. Controlled by the customer. Neither completely trusted nor completely untrusted. Public accessible services will be positioned and deployed here.

Customer controlled firewalls are placed between the external zone and the DMZ and between the DMZ and the internal zone. Some public services of the DMZ can usually be reached from the external zone, while in a security puristic form the the internal zone is strictly inaccessible from the external zone. Tightly controlled access from the DMZ to the internal zone may be permitted for specific services or might be completely forbidden. Access from the internal zone to the DMZ or external zone and from the DMZ to the external zone will be generally permitted, but may also be restricted in a customer specific way.
This can lead to a deployment dilemma, when a site is on the one hand obliged to a very strict security policy in the form that the DMZ can under no circumstances access the internal zone, while on the other hand it wants to deploy workflow services that span the internal and the external zone in terms of participants involved or services used.
Placing the workflow service at the internal zone is clearly unfeasible, since external access is strictly forbidden. While placing the workflow service in the DMZ would ensure policy compliant access from the external as well as from the internal zone, it would also expose a central service completely in the DMZ, raising the potential threat level. Access to internal services would be still forbidden.
As a consequence, the logical workflow service has to be partitioned into an internal server and an externally accessible ("external") server. The internal server is placed in the internal zone, solely accessible from there, the external server is placed in the DMZ, ensuring accessibility from the external zone.

A special coordination/communication pattern ensures controlled connectivity between the two servers. The internal server can initiate communications to the external server, but in accordance to the security policy, the external server cannot initiate connections to the internal server/zone. Messages for the internal server are buffered and kept at the external server. The internal server will periodically contact the external one and actively fetch the messages destined for it.

7.1.2 General solution elements

We will itemize the elements and properties of the solution architecture:

- *System layout:* separate the external part of the workflows from the internal processing.
  - there will be an external workflow server placed in the DMZ
  - there will be an internal workflow server placed in the internal zone
  - both servers have their own DBMS deployed in the same zone as the server itself
- *Connectivity and network traffic:* no network traffic to the internal server is to be initiated by the external server.
  - all communication is to be initiated by the internal server
  - externally arising needs to communicate with the internal server must be queued at the external server
  - the internal server will periodically poll the external one
  - communications from the internal server to the external one can be done without buffering
  - network or server outages can arise, reliable messaging from the internal server to the external server requires the ability to also buffer messages internally and resend them when the network issues are fixed.
- *Workflow participants:* are to be separated into two classes:
  - external participants are only allowed to access the external server
  - internal participants work solely at the internal server
  - participant assignment to servers / zones is accomplished via roles or rights
- *Master Data:* is maintained at the internal server
  - a subset of the master data (like participants, rights, organizational structures, ...) is to be replicated from the internal server to the external one
  - the subset is specified at the internal server
  - can be replicated periodically via a timer or replication can be initiated by the administrator
- *Process Definitions:* must be adapted with respect to the following aspects

  - manual separation of the logical process flow into external and internal parts needed
  - process data design must also take into account used/needed relationships with respect to server/zone
  - definition of handover points in control flow between internal and external servers (and vice versa)
  - handover points are implemented as manual activities, specific details are stated in a separate XML file
  - this approach has low impact on local process execution, it decouples the local from the remote process execution

- *Process execution:*
  - a logical process can be started either internally or externally, not at both servers
  - a process instance started externally
    - can start an internal process instance
    - wait for changes in the internal process
    - continue, when the internal process reaches a particular step (handover point)
    - receive updated process instance data (forms/documents)
  - a process instance started internally
    - can start one external process instance
    - wait for changes in the external process
    - continue, when the external process reaches a particular step (handover point)
    - receive updated process instance data (forms/documents)

- *Timers:*
  - *PerimeterReplicationTimer*: is used for master data replication; must be activated on the internal server if timer based replication is desired.
  - *PerimeterSyncTimer:* is used for remote process creation and information about status changes. It interprets the handover specification files *externtask.xml* and *interntask.xml*.
  - *WfXMLTask*: assures message delivery of buffered messages. Messages will be buffered either when network/server outages occur or will be buffered on principle at the external server, since initiation of network operations is prohibited.

- *Data aspects:*
  - data is transferred back and forth along with the center of control
  - process forms and documents are transferred
  - a subset of data eligible for transfer can be specified
  - data design must take into account external/internal separation
  - referenced non process data must be reachable in the context (must e.g. be replicated via master data replication)

7.2 Examples of logical process design and process separation

We will give two examples for zone-spanning workflow processes which differ in complexity and center of control.

7.2.1 Single step external processes (multi incarnations)

Consider a simple process

```
process PSimple
begin /*internal*/
   intern step_I1;
   extern step_E1;
   intern step_I2;
   intern step_I3;
end
```

The process is started internally, when *step_I1* is finished, the process should continue on the external server with *step_E1*, after finishing this step the process should be transferred back to the internal server and continue in *step_I2*.
By separating this logical process into an internal and an external one we arrive at:

```
process PSimpleExt
begin
   extern step_E1;
end  // implicit handover
```

```
process PSimpleInt
begin /*internal*/
   intern step_I1;
   extern step_E1;  // handover
   intern step_I2;
   intern step I3;
end
```

The process separation is rather straightforward, the external process consists of a single step (it is a 'mini-process'). The internal process remains virtually unchanged. There is one internal process instance, and one external process instance for each incarnation of *step_E1*. Finishing the external process means that control is given back to the internal process. For each new execution of *step_E1* (e.g. via going back) where will be a new external process instance.

7.2.2 Interleaved internal and external processes

The following process is somewhat more complex, the control flow starts at the external server and is transferred to the internal server and back several times. The loop also implies that this can be repeated several times:

```
process PInterleaved
begin /*external*/
```

```
  all step_E0;
  repeat
    extern step_E1;
    intern step_I11;
    intern step_I12;
    extern step_E21;
    extern step_E22;
    intern step_I21;
    intern step_I22;
    extern step_E3;
  until finished()
  extern step_E4;
end
```

After the manual separation of the logical process we get two process fragments:

```
process PInterleavedExt
begin /*external*/
   all step_E0;
   repeat
      extern step_E1;
      intern step_I1;  // handover
      extern step_E21;
      extern step_E22;
      intern step_I2;  // handover
      extern step_E3;
   until finished()
   extern step_E4;
end // implicit handover
```

```
process PInterleavedInt
begin /*internal*/
   repeat
      intern step_I11
      intern step_I12;
      extern step_E2; // handover
      intern step_I21;
      intern step_I22;
     extern step_E3_E1; // handover
   until finished()
end
```

In the external process, each logically external step is included unchanged and for each sequence of internal steps, an artificial step is inserted instead of the sequence.
Vice versa, in the internal process each logically internal step is included unchanged and for each sequence of external steps, an artificial step is inserted instead of the sequence.
Process separation is more complex and can be challenging, depending on the control structures of the processes.
There will be one internal and one external process instance, regardless of the number of loop iterations. The center of activity will be in just one of the instances while the other instance is waiting.

7.3 Configuration of the servers

7.3.1 Basic Installation

Internal Server

The internal server must be installed first. There are no special considerations concerning the basic installation, except that server number 1 must be used. The server number (parameter *avw.servernumber* in avw.conf) must be defined at setup wizard (step 2) before the database schema is created! The id of the server object must have the suffix *_intern*[1]
Ensure that email communication (SMTP) is enabled and working. After basic installation, the following master data objects must be created:

- a right with id *extern*: will later be assigned to agents who will work with the external server.
- a right with id *intern*: will later be assigned to agents who will work with the internal server.
- a user with id *wfxml*: this is a system user, tasks being transferred to the other server or being executed there will have this user as agent. This user must be given universal rights *Create Objects* and *Edit Objects*.
- a role with id *troubleshooter*: the WfXML subsystem might forward problematic process instances to members of this role (each role type is possible).
- a (process) form with id *troublenote*: to transport details about problematic instances. Must have one "subject" field (250 chars) and one "content" field (4000 chars). Please note that checkbox *Usable in DMS* must be activated for this form type!

External Server

The basic installation of the external server is special in one critical way: in order to assure disjoint ranges of oid values, a vastly different server number must be used. We suggest at least 65536 [2]. The server number (parameter *avw.servernumber* in avw.conf) must be defined at setup wizard (step 2) before database schema is created!
The id of the server object must have the suffix *_extern*.

7.3.2 WfXML Configuration

Internal Server

- At Administration/Configuration/Communication/Enable Wf-XML : set to *Active*. Leave other WfXML related properties as they are.
- At Administration/Admin tasks/Communication/WfXML/Partner list: enter a new WfXMLPartner (the external server) with the following information:

[1] If your server was already up and running before and you want to change your (up to now single) server to be the internal one, ensure that the server name is being changed at Administration/Configuration/Cluster/Server name and do also change the id of the server object in Administration/Admin Tasks/Cluster/Servers.

[2] The initial oid starts with $2^{32} * servernumber$.

  - Server: id of the external server; must have the suffix *_extern*.
  - Operating mode: *Passive*
  - Protocol, Hostname, Port: the Http address components of the external server.[3]
  - Path: use *<ctx>/servlet.method/com.groiss.wfxml.impl.Receiver.receive*

- At Administration/Admin tasks/Server/Timers : activate the *WfXMLTask timer*, choose a sensible interval.

- Put the two following lines in the avw configuration (section "Other parameters"):

  ```
  avw.wfxml.outgoingmessagemodifier=com.groiss.perimeter.ExternHandler
  avw.wfxml.exceptionhandlers=com.groiss.perimeter.ExternHandler
  ```

  Those entries take care of subject / duedate updates and of exception handling via the *troubleshooter* role.

External Server

- At Administration/Configuration/Communication/Enable Wf-XML : set to *Passive*. Leave other WfXML related properties as they are.

- At Administration/Admin tasks/Communication/WfXML/Partner list: enter a new WfXMLPartner (the internal server) with the following information:

  - Server: id of the internal server; must have the suffix *_intern*.
  - Operating mode: *Active*
  - Protocol, Hostname, Port: the Http address components of the internal server.
  - Path: use *<ctx>/servlet.method/com.groiss.wfxml.impl.Receiver.receive*

- At Administration/Admin tasks/Server/Timers : activate the *WfXMLTask timer*, choose a sensible interval.

- Put the two following lines in the avw configuration (section "Other parameters"):

  ```
  avw.wfxml.outgoingmessagemodifier=com.groiss.perimeter.ExternHandler
  avw.wfxml.exceptionhandlers=com.groiss.perimeter.ExternHandler
  ```

  Those entries take care of subject / duedate updates and of exception handling via the *troubleshooter* role.

Network Infrastructure

Configure your firewall between the internal and the external zone so that HTTP(S) requests from the internal server IP to the external server IP and port are permitted.

[3] At the moment, HTTPS is not supported. It may be available at a later time.

7.3.3 Master Data Synchronization

As already mentioned, master data is maintained on one server (the internal one) and replicated to the other server. This is done via setting up of a replication channel:

Internal Server

At Administration/Admin tasks/Communication/WfXML/Replication Channel, add a new Replication Channel:

- *Id:* use something with a suffix like *_to_extern*
- *ReplicationPartner:* select the external server
- *Direction:* set to *outgoing*
- *Active:* check it
- *Check with Timer:* check, if master data should be replicated by the timer; leave unchecked, if it should be initiated manually.
- *Classes:* list of Java Object Class names of those classes which should be replicated. The class path check icon can be used to see the effective list. If a class name is prefixed by $-$, then it is removed from the list. $-*$ can be used to start with an empty list and not with the default one.

If the replication should be done by the corresponding timer, then go to Administration/Admin tasks/Server/Timers and activate the *PerimeterReplicationTimer*, choose a sensible interval or cron pattern.

On the mask *Replication Channel*, the replication metadata (oid and time stamp of the last replication) can be seen. The replication operation for a replication channel can be initiated via the provided *Start replication* button. The button *Show partner state* can be used to see the corresponding info on the partner (the other server). The *Synchronize Replication Info* button can be used to synchronize the replication metadata. This does only work from an *active* server (the internal one).

Hint: Following elements are not synchronized automatically with *PerimeterReplication-Timer* or button *Start replication*:

- Form templates (html files) in forms directory
- Password of user objects

External Server

At Administration/Admin tasks/Communication/WfXML/Replication Channel, add a new Replication Channel:

- *Id:* use the same id as the replication channel at the internal server.
- *ReplicationPartner:* select the internal server

- *Direction:* set to *incoming*
- *Active:* check it
- *Check with Timer:* do not check it
- *Classes:* leave it empty

7.3.4 Process definitions

We will sketch the deployment of process definitions along the example of the interleaved process mentioned in section 7.2.2. The handover specifications files are given and will be explained.

Internal Server

- deploy the forms used in *PSimpleInt* and *PInterleavedInt*
- deploy the process definition *PSimpleInt* and*PInterleavedInt*; for external steps agents (user or role) with permission *extern* must be defined, for internal steps agents with permission *intern* (see section 7.3)
- activate the *PerimeterSyncTimer* timer and give it a sensible interval
- create the handover file *externtasks.xml* and put it into classpath:

```
<?xml version="1.0"?>
 <ExternTasks>
  <Process appl="default" id="PSimpleInt" version="any">
   <Task id="step_E1" operation="start">
    <RemotePartner id="server_extern"/>
    <RemoteProcess appl="default" id="PSimpleExt" version="1"/>
    <Input>
     <ProcessForm id="f"/>
     <DMSDocuments/>
    </Input>
   </Task>
  </Process>
  <Process appl="default" id="PInterleavedInt" version="any">
   <Task id="step_E2" operation="inform">
    <Input>
     <ProcessForm id="f"/>
     <DMSDocuments/>
    </Input>
   </Task>
   <Task id="step_E3_E1" operation="inform">
    <Input>
     <ProcessForm id="f"/>
     <DMSDocuments/>
```

```
    </Input>
   </Task>
  </Process>
 </ExternTasks>
```

It consists of a single element *ExternTasks* and must obey the DTD found in file *ep.jar* at *classes/conf/perimetertasks.dtd*. Within it, there will be one *Process* element for each of the processes definitions which are to be observed in order to implement the perimeter communication pattern. The attributes of the *Process* element specify the id of the application, the id and version of the process definition. For each handover point, a *Task* element gives the details. The *id* of the step must be specified and an *operation* is to be given. The following operations are allowed:

- *start*: states that a new process instance is to be started at the other server, in the nested *RemotePartner* element, the *id* of the WfXML partner object must be given, and the nested element *RemoteProcess* states the process to be started via attributes *appl*, process *id* and *version*.
- *inform*: states that the corresponding process instance on the other server is to be continued. Remote partner and process are implicitly known.
- *start_or_inform*: this is a combination of the *start* and *inform* operations, it will be used in loops. The first iteration will lead to a remote process start, the next iterations will merely inform the other already existing process instance.

Each *Task* element has a nested *Input* element where process forms (via the id of form variable) and documents to be transferred are stated [4].
The given file states the three handover points from the internal to the external server according to the two process definitions sketched above.

External Server

- deploy the forms used in *PSimpleExt* and *PInterleavedExt*
- deploy the process definition *PSimpleExt* and *PInterleavedExt*
- activate the *PerimeterSyncTimer* timer and give it a sensible interval
- create the file *interntasks.xml* and put it into classpath:

```
<?xml version="1.0"?>
 <InternTasks>
  <Process appl="default" id="PInterleavedExt" version="any">
   <Task id="step_I1" operation="start_or_inform">
    <RemotePartner id="server_intern"/>
    <RemoteProcess appl="default" id="PInterleavedInt" version="1"/>
    <Input>
     <ProcessForm id="f"/>
```

[4] Details for the specification will be given in a future version of this document

```
        <DMSDocuments/>
      </Input>
    </Task>
    <Task id="step_I2" operation="inform">
      <Input>
        <ProcessForm id="f"/>
        <DMSDocuments/>
      </Input>
    </Task>
  </Process>
</InternTasks>
```

The handover specification at the external server is called *interntasks.xml*. Within its single *InternTasks* element, the handover points to the internal server are specified.

8 @enterprise *and Datasources*

This chapter describes the configuration of datasources in **@enterprise** and gives example configurations for *Tomcat* and *Jetty 6.1*.
Before version 8.0, **@enterprise** could use the traditional method to acquire connections to the database via the DriverManager. From version 8.0 and onward, datasources are an alternative way to obtain database connections.

8.1 Configuration of a Datasource in @enterprise

To use a datasource in **@enterprise**, the JNDI-path of the datasource must be specified instead of the JDBC-URL. Instead of e.g.

jdbc.derby://localhost:1527/ep;create=true

use something like

jdbc/DerbyDB

If the datasource path starts with './', it will be looked up in the initial JNDI context (without the './' prefix). If not, it will be looked up in the 'java:/comp/env/' subcontext of the initial JNDI context. When using the datasource it is not needed to provide a JDBC-driver or to fill in the following configuration items:

- Database Userid
- Database Password

The other database related configuration items are still needed and used.

8.2 Configuration of a Datasource in Tomcat

This section describes how Tomcat can be configured:

1. Put the JAR-file of the JDBC-driver into the *lib* directory of Tomcat.
2. Deploy the **@enterprise** WAR-file (e.g. using ep90 as the contextpath)
3. Go to *../conf/*<*service*>/<*host*> directory and put a <contextpath>.xml file there.

- <*service*>*:* The name of the Tomcat service (as in the service element in the ../conf/server.xml file); usually Catalina
- <*host*>*:* The name of the Tomcat host (as in the host element in the ../conf/server.xml file); usually localhost
- <*contextpath*>*:* The contextpath where **@enterprise** is deployed

So, you would end up with a file named ../conf/Catalina/localhost/ep90.xml. In this file specify the datasource as a resource within the context element:

```
<Context>
  <Resource
  name="jdbc/DerbyDB"
  auth="Container"
  type="javax.sql.DataSource"
  factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
  maxActive="12"
  username="derby"
  password="derby"
  driverClassName="org.apache.derby.jdbc.ClientDriver"
  url="jdbc:derby://localhost:1527/ep;create=true"
  />
</Context>
```

The value of the *name* attribute must match the path of the datasource in the **@enterprise** configuration file. The value of attribute *factory* can be also org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory depending on your Tomcat version.

Details for the other parameters can be found in the Tomcat documentation.

4. The following step may not be needed. Include the reference to the resource in the *web.xml* descriptor of **@enterprise** application:

```
<resource-ref>
    <description>DB Connection</description>
    <res-ref-name>jdbc/DerbyDB</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

The content of the *resource-ref-name* element must match the path of the datasource in the **@enterprise** configuration file.

5. Restart Tomcat, start the **@enterprise** application and begin to setup **@enterprise**.

8.3 Configuration of a Datasource in Jetty 6.1

This section describes the configuration of **@enterprise** running as web-application in a jetty-installation (**@enterprise** does not start jetty as embedded web-server!):

1. Create a *myjetty.xml* file that activates the needed jetty-plus features. Using the *jetty.xml* and *jetty-plus.xml* files as model. Add the following lines to the server configuration element:

```
<Array id="plusConfig" type="java.lang.String">
        <Item>org.mortbay.jetty.webapp.WebInfConfiguration</Item>
        <Item>org.mortbay.jetty.plus.webapp.EnvConfiguration</Item>
        <Item>org.mortbay.jetty.plus.webapp.Configuration</Item>
        <Item>org.mortbay.jetty.webapp.JettyWebXmlConfiguration</Item>
        <Item>org.mortbay.jetty.webapp.TagLibConfiguration</Item>
</Array>
```

   Add the configuration classes also to the *addLifeCycle* call element:

```
<Call name="addLifeCycle">
 <Arg>
  <New class="org.mortbay.jetty.deployer.WebAppDeployer">
   <Set name="contexts"><Ref id="Contexts"/></Set>
   <Set name="webAppDir">
    <SystemProperty name="jetty.home" default="."/>/webapps
   </Set>
   <Set name="parentLoaderPriority">false</Set>
   <Set name="extract">true</Set>
   <Set name="allowDuplicates">false</Set>
   <Set name="defaultsDescriptor">
    <SystemProperty name="jetty.home" default="."/>/etc/webdefault.xml
   </Set>
   <Set name="configurationClasses"><Ref id="plusConfig"/></Set>
  </New>
 </Arg>
</Call>
```

2. Uncompress the **@enterprise** WAR-file (e.g. using ep90 as contextpath).

3. Put the JAR-file of the JDBC-driver into the *lib* directory of the web-application.

4. Go to the *../webapps/ep90/WEB-INF* directory and put a *jetty-env.xml* file there. In this file specify the datasource as a resource within a context element:

```
<Configure class="org.mortbay.jetty.webapp.WebAppContext">
 <New id="DerbyDB" class="org.mortbay.jetty.plus.naming.Resource">
   <Arg>jdbc/DerbyDB</Arg>
   <Arg>
    <New class="org.apache.derby.jdbc.ClientDataSource">
          <Set name="databaseName">ep</Set>
          <Set name="portNumber">1527</Set>
```

```
            <Set name="serverName">localhost</Set>
            <Set name="user">derby</Set>
            <Set name="password">derby</Set>
        </New>
      </Arg>
    </New>
  </Configure>
```

The value of the first *Arg* element must match the path of the datasource in the **@enterprise** configuration file.

5. Restart Jetty with following parameter and begin to setup **@enterprise**:

   java -jar start.jar etc/myjetty.xml

8.4 Considerations for pooled Datasources

@enterprise still uses its own connection pool, even when the datasource is a pooled one. We have better control over the connection this way, and can provide all features of the **@enterprise** pool itself (session environment, automatic reconnect, ...).
This strategy imposes two requirements for a pooled datasource:

- It should never expect to get the connection back or destroy connections in use. In a DBCP connection pool, this can be implemented via

  removeAbandoned="false"

  In a Weblogic pooled datasource this can be achieved via disabling the "Inactive Connection Timeout" feature by setting it to 0.

- The pool size should be large enough to provide the max. number of connections specified in the **@enterprise** configuration (see chapter 3) increased by at least 2 (for internal connections used by the engine itself).

A Database Performance Hints under Oracle

A.1 Preliminaries

The statements in this chapter refer to an **@enterprise** installation with an Version 8 Oracle DBMS. It is assumed that no atypical characteristics concerning either data distributions or data volumes or transaction volumes like extremely long worklists or BLOBs dominate the system. Further we assume that no other significant workload besides the **@enterprise**-service is processed on the system (dedicated hardware).

For successful performance improvements, the most crucial issue is to correctly identify and pinpoint system bottlenecks. Applying tuning actions without having a specific hint about the kind or reason for unacceptable performance is not target-oriented. It is essential to isolate and contain the problem area (database, **@enterprise** server, CPU, memory, network, own application classes, specific user operations). One should apply all means and tools which are offered by the underlying platform to check performance parameters or monitor them on a regular basis. Because of the wide variety of the platforms concerning this specific area, we refer the reader to the appropriate systems documentation.

We assume that the reader has some basic familiarity about the architecture of Oracle and is somewhat acquainted with its significant mechanisms.

A.2 Key Operating Parameters of the Database

The following parameters are vitally important for an efficient operation of the database. They all can be found in the **ini.ora** file.

DB_BLOCK_SIZE States the size of the data blocks in the DB. In most environments the default value is 2048 bytes. For **@enterprise** the value should be increased to 4096 or 8192. The change should reduce IO-overhead and has no other significant implications. Unfortunately, the value can't be changed in an existing data base, one would be forced to apply a complete export/import cycle to apply a modification.

DB_FILE_MULTIBLOCK_READ_COUNT Determines how many blocks are read during a full table scan. The value should be dimensioned in such a way, that the product of DB_BLOCK_SIZE and DB_FILE_MULTIBLOCK_READ_COUNT equals the size of the operating system buffer (often 64K). The value can be changed during operations but is applied only at the next startup of the database instance.

DB_BLOCK_BUFFERS States the size of the database block buffer caches in units of blocks. It is an extremely crucial parameter. The default values of Oracle are way to small. For an application system with the characteristics of **@enterprise** (mostly interactive users in OLTP, insignificant batch processing) one should configure the cache size to achieve a hitrate above 95% to 98% in regular operations. Regular monitoring is essential. One could apply the following queries (as user SYSTEM) to determine current hit rates:

```
select
  SUM(DECODE(Name, 'consistent gets',Value,0)) Consistent,
  SUM(DECODE(Name, 'db block gets',Value,0)) Dbblockgets,
  SUM(DECODE(Name, 'physical reads',Value,0)) Physrds,
  ROUND(((SUM(DECODE(Name, 'consistent gets', Value, 0))+
    SUM(DECODE(Name, 'db block gets', Value, 0)) -
    SUM(DECODE(Name, 'physical reads', Value, 0)) )/
     (SUM(DECODE(Name, 'consistent gets',Value,0))+
      SUM(DECODE(Name, 'db block gets', Value, 0))))
      *100,2) Hitratio
from V$SYSSTAT;
```

```
column HitRatio format 999.99
select Username,
      Consistent_Gets,
      Block_Gets,
      Physical_Reads,
      100*(Consistent_Gets+Block_Gets-Physical_Reads)/
         (Consistent_Gets+Block_Gets) HitRatio
 from V$SESSION, V$SESS_IO
where V$SESSION.SID = V$SESS_IO.SID
  and (Consistent_Gets+Block_Gets)>0
  and Username is not null;
```

If an unsatisfactory hit rate is measured, DB_BLOCK_BUFFERS should be increased in steps of 15% to 25%, until hit rate levels out. Meaningful measurements are only possible in real production mode and not immediately after the startup phase of the instance when the cache is still cold.
It is common knowledge, that the buffer cache should not be increased beyond certain thresholds. Each word of main memory that is allocated exclusively for the buffer cache can be in high demand by other system components. In no way the machine should be

pressed to swapping or paging activities. After every expansion of buffer cache size, measurements with a warm cache are called for in combination with keeping an eye on paging or thrashing. Memory expansions should be considered at such points.

SHARED_POOL_SIZE Determines the size of the shared pool in the System Global Area (SGA). Oracle defaults are often found to be too small.
A rule of thumb says that 15% to 20% of the shared pool should stay free.
The current size can be calculated as follows:

```
select value from v$parameter where name='shared_pool_size';
```

The free space is returned by this query:

```
select name, bytes from v$sgastat where name='free memory';
```

Key elements in the shared pool are the library cache and the data dictionary. Miss rates for both components can be determined with the help of the following queries. In the library cache miss rates of under 1% and of under 5% in the data dictionary are commonly seen as appropriate.

```
column "Executions" format 9,999,999,990
column "Cache Misses Executing" format 9,999,999,990
column "Data Dictionary Gets" format 9,999,999,999
column "Get Misses" format 9,999,999,999
column "% Ratio" format 999.99



select sum(pins) "Executions",
     sum(reloads) "Cache Misses Executing",
     (sum(reloads)/sum(pins)*100) "% Ratio"
from v$librarycache;


select sum(gets) "Data Dictionary Gets",
     sum(getmisses) "Get Misses",
     100*(sum(getmisses)/sum(gets)) "% Ratio"
from v$rowcache;
```

If higher mis rates are measured, we advise a similar procedure like in the case of the DB_BLOCK_BUFFERS parameter.

SORT_AREA_SIZE Size of the area in the main memory which is reserved for each user for in-memory sorting operations. If disk-based sorts make up for more than 5% to 10% of the in memory sorts, then SORT_AREA_SIZE should be increased. The current configuration can be determined with:

```
select substr(name,1,25) Name,
       substr(value,1,15) Value
  from V$PARAMETER
 where Name = 'sort_area_size';
```

Statistics about the number of sorts, separately for main memory and disk based sorts are implemented by:

```
select substr(name,1,25) Name,
       substr(value,1,15) Value
  from V$SYSSTAT where name like 'sort%';
```

LOG_BUFFER Size of the redo log buffer in the SGA.
The current size can be obtained by:

```
select substr(name,1,25) Name,
       substr(value,1,15) Value
  from V$SGA
 where Name = 'Redo Buffers';
```

If redo log space requests are issued in the database, there might be a bottleneck here. The following query investigates this:

```
select  substr(name,1,25) Name,
       substr(value,1,15) Value
from v$sysstat
where name = 'redo log space requests';
```

The value should approximate zero. If this is not the case, one should increase the LOG_BUFFER parameter in steps of 50% to 100%. It might be advisable to increase the shared pool size by the same (absolute) amount.

A.3 Optimizer

Cost based optimization is the way to go with Oracle. In general, better query plans can be generated than pure rule based optimization could achieve.
To activate the cost based optimizer, the parameter OPTIMIZER_MODE in init.ora must be set to CHOOSE. It is also necessary to statistically analyze the data distribution and index selectivity.
Oracle offers commands of the form analyze table <mytable> compute statistics. One can supplement statistics for an entire schema using
execute dbms_utility.analyze_schema('USER','COMPUTE');. The 'USER' element should be replaced by the name of the **@enterprise** data base user.
It is highly advisable to run this command from time to time. In any case, it should be run periodically during the first period of production use and additionally when significant

configuration changes (new applications, other data volumes) take place. The analysis is quite resource intensive and should not be applied during peak operational hours. Sufficient temporary tablespace must be provided, also. A practical trade-off between statistical accuracy and resource consumption can be achieved through use of 'ESTIMATE' instead of 'COMPUTE'. In this case the system takes samples of the data and does not go through the entire volume. A good strategy might be to establish a batch-job which issues this schema analysis commands on a regular (weekly) basis.

A.4 Storage

A.4.1 Disks

The main performance issues in the disk subsystem are the separation of random access and sequential access and further to isolate individual sequential accesses.
More precisely, separate the redo-logs, the after image files and the rollback segments, and put them on individual disks without any further activity.
Further split up SYSTEM and TEMPORARY tablespaces from the rest of the system.
Tables with particular high activity on them are AVW_STEPINSTANCE, AVW_FOLLOWS and AVW_FORMVERSION. A good measure would be to place them together with their indices on separate tablespaces, to be able to place them on specific disks and to distribute the load on multiple devices. Another possible strategy would be the division of index space and table data space in different tablespaces.
It is not possible to give general advice without deeper knowledge of the operational characteristics. Nevertheless, for an installation with significant size, we strongly recommend to devote some thoughts to this issues and to divert from the default configuration.
An overview about IO distribution over the individual data files can be gained by:

```
select DF.Name File_Name,
     FS.Phyblkrd Blocks_Read,
     FS.Phyblkwrt Blocks_Written,
     FS.Phyblkrd+FS.Phyblkwrt Total_IOs
 from V$FILESTAT FS, V$DATAFILE DF
where DF.File#=FS.File#
order by FS.Phyblkrd+FS.Phyblkwrt desc;
```

A.4.2 Parameters for Tablespaces

Appropriate default storage parameters for the tablespaces would be:

```
alter tablespace AVW default storage
  (initial 256k next 256k maxextents 200 pctincrease 0);
```

Instead of AVW, state the tablespaces which are used to store the **@enterprise** tables and indexes, in particular the default tablespace of the **@enterprise** database user. For some tables which can be assumed to have a greater size than that (50MB) like AVW_STEPINSTANCE,

AVW_FOLLOWS and AVW_FORMVERSION, the storage parameters can be changed in full operation mode; e.g.:

```
alter table <mytable> storage(next 1M maxextents 1200);
```

With this statement, table <mytable> can use 1000 additional extents, each being 1 MB in size when one assumes that 200 extents were already used. It is generally advisable to use zero as value for pctincrease, to avoid exponentially increasing storage demand for extents.

A.5 One owns Tables and Queries

For own tables which are used to store application relevant data, exactly the same considerations like for system tables according to table placement and to storage parameters should be made. In particular, popular access paths should be supported by appropriate (multi-column) indexes.
Queries of application tables should generate a result set as small as possible. It is recommended to use a two phase approach for queries with potentially large result sets. First, the number of tuples (count(*)) should be determined. If this number exceeds a certain threshold, it is time to give the user a chance to decide upon further execution of the query. The user could apply additional constraints to the search condition which would further confine the result set, or she could explicitly get the whole large result set (and thereby accepting higher response time and workload on the server).
For medium sized tables, which are often scanned in their entirety, table level caching could be advantageous:

```
alter table mytable cache;
```

Clearly, sufficient space in form of DB_BLOCK_BUFFERS must be provided.
Criteria in queries should be used in such a way that indexes get used. Strive for point queries or at least for multipoint queries with high selectivity. (its better to use a='b' than a like 'b%' which is in turn better than a like '%b%').
Of uttermost importance is the usage of the @**enterprise** transaction cache mechanism, which works for all subclasses of SQLObject. Access to such objects should be done through receiver.get(oid) and not via receiver.get('oid=xxx');.
Performance friendly formulation of application queries (especially such statements which are executed quite often) call for generation, interpretation and perhaps modification of the execution plans. Measures could be the definition of additional indices or clustering on a physical level or semantic preserving reformulation of the query or explicit incorporation of query optimization hints.
Concerning these issues we refer to the 'Oracle8 Tuning' and 'Oracle8 Concepts' and 'Oracle8 Application Developers Guide' manuals. Consider the possibilities of TKPROF and EXPLAIN PLAN. The logfile of @**enterprise** may have valuable first hints like duration of SQL statements.
It is much better to run complex queries in their entirety on the DB-server than to overflow the server with lots and lots of simple individual queries and to stick their results together in the @**enterprise** server. This is due to relatively high startup and communication overhead and context switches between the two servers.