Server System Administration Guide

Installation
Update
Maintenance

ScientificCMS
System Administration Series

Server System Administration Guide

Version 2.0

This guide contains instructions and reference information for system administrators to install, update and maintain the ScientificCMS software. It is focused on Red Hat Enterprise Linux or Fedora distributions but may be used for other Linux distributions or Unix operating systems as well. It also contains some hints for working on the Windows operating system.

ScientificCMS:
Server System Administration Guide Release 2.0

Release 2.0.0 (2013-04)

edited by: Peter Boy, University of Bremen (pb@zes.uni-bremen.de)

Acknowledgments

This guide uses material from administration manuals for previous versions of Red Hat's documentation for the Collaboration and Content Management system (CCM) and APLAWS+, a system specifically tailor-made for Great Britain Local Authorities based on CCM. Large parts had been written by Oliver Sharpe (Red Hat) and Arturo Dell (Camden). It also contains information, gathered from various contributions of users, namely on discussion forums and wikis.

Document History:

Version	Date	Origin	Written by	

Table of Contents

Preface

What is ScientificCMS?

ScientificCMS is an open source application that makes the creation and maintenance of websites easier for academic research institutions, while at the same time facilitating scholarly communication and collaboration.

ScientificCMS shares the codebase and cooperates closely with the *APLAWS+* project, a CMS that was designed for local governmental authorities in the United Kingdom. Both projects are user-group-specific adaptations of (Open)CCM, a general Web content management framework, which was originally developed at Massachusetts Institute of Technology by Philip Greenspun and others.

Scope of This Guide

This guide contains step by step instructions as well as reference information for system administrators to install, update and maintain the ScientificCMS software.

The descriptions are focused on Red Hat Enterprise Linux or Fedora Linux distributions but may be used for other Linux distributions or Unix operating systems, as well. On Windows operating systems the necessary steps are very similar and can easily be performed accordingly. Some hints are given, nevertheless.

Part I covers the installation of the software. It provides an conceptual overview as well as step by step instructions of the installation process. In the examples we assume an installation on a standalone server for sake of simplicity. Part III – maintenance – describes steps to extend the installation running ScientificCMS in a cluster of front end systems for *high availability* and *load balancing*, served by a back end database server (multi tier environment).

Part II is reserved to cover the process of updating a ScientificCMS installation. Currently version 2 is the first publicly available version of ScientificCMS, so there is no need to update any installation.

Part III deals with issues of maintenance and deployment. Specifically it describes various ways of integrating ScientificCMS into an Apache environment, the distribution of individual components on different servers, performance tuning, and other deployment issues.

Intended Audience

This guide is written with a system (server) administrator in mind. He or she needs no knowledge about the internals of ScientificCMS. All tasks are performed on operating system level. Instead administration privileges on the server machine are required as well as a sound knowledge about working with a text editor, operating system commands and editing configuration files.

Release Information

This guide is specifically based on **version 2.0** of ScientificCMS but should be applicable for all 2.x versions. As soon as changes or new functions are included, a new version will be released.

System Administration Series

The System Administration Series provides information for the server administrator about installing and maintaining the software. You need root authority, but no knowledge about the ScientificCMS collaboration and content management system.

Previous Versions of This Guide

This guide is based on several manuals created by Red Hat for the CCM framework as well as by the sister project, APLAWS+. It updates the information and adds a description of procedures and functionality that were missing there. Specifically it supersedes:

- Red Hat WAF Installation Guide
- APLAWS+ Installation Guide Version 1.0.4
- LAWs Quick Start Install Guide

Part I
Installation

This part describes the basic installation of ScientificCMS including the most important post-installation steps.

It provides a conceptual overview as well as step by step instructions of the installation process. After completion you should have a standalone server up and running that can serve both as a production system, as well as a basis for deploying a complex multi-server environment.

2: For the Hurried Administrator: Quick Installation Guide

2.1 Installing an Application Program Edition

1. Ensure Java 1.6 (JRE is sufficient) is installed (1.5 should do as well).
2. Ensure PostgreSQL 8.x or Oracle SE is installed and running .
3. Create a dbms user for (e.g. ccm) and a database (e.g. ccm). In case of postgresql additionally execute "`createlang plpgsql ccm`".
4. Acquire root permission and install the ScientificCMS Application Program installation file, e.g. "`rpm -i scientificcms-2.0-std.rxyz.noarch.rpm`"
5. Switch temporarly into account yy and make `/usr/share/scientificcms/webapps/ROOT/WEB-INF/bin` as default.
6. Execute "`sh ccm prepare`" and fill in the missing bits of information.
7. Optionally use "`sh ccm set`" to adjust configuration parameters.
8. Optionally adjust the default memory allocation in `/etc/sysconfig/scientificcms` (recommended `-Xmx1024m` or higher depending on usage)
9. Start Tomcat

2.2 Installing WAR File Edition

Perform the following steps:

1. Ensure Java 1.6 (JRE is sufficient) is installed (1.5 should do as well).
2. Ensure PostgreSQL 8.x or Oracle SE is installed and running .
3. Create a dbms user (e.g. ccm) and a database (e.g. ccm). In case of postgresql additionally execute "`createlang plpgsql ccm`".

4. Install Tomcat 6 (or a compatible servlet container) according your operating system instructions.
5. Optionally adjust the default memory allocation in `/etc/sysconfig/scientificcms` (recommended -Xmx1024m of higher depending on usage)
6. Don't start Tomcat and stop Tomcat if it is started by the installation program.
7. Unzip the war file into `~/webapps/ROOT`
8. Make `~/WEB-INF/bin` as default.
9. Execute "`./ccm prepare`" and fill in the missing bits of information.
10. Optionally use "./`ccm set`" to adjust configuration parameters.
11. Start Tomcat

3 Application Bundles

ScientificCMS consists of several independent, but highly integrated modules, where each module handles either a specific task (application, e.g. the content management itself, or a discussion forum, or an online query) or a specific type of information (content type, e.g. article, job offer, legal notice, etc.). For detailed information consult the Content Administration Guide.

Currently ScientificCMS includes 55 modules. For ease of installation there are 2 different “bundles”, i.e. preconfigured sets, which cover typical installation needs. This chapter outlines their differences.

Currently you have to decide between 2 bundles for a production server:

- **Base** Bundle
- **Standard** Bundle

There is another bundle available, **Demo** Bundle. It installs the same set of modules as the Standard Bundle, additionally with a preset configuration. It is not meant for production, as the name suggests.

3.1 Standard Bundle

This bundle contains the recommended set of applications to get an ScientificCMS server operational. The set of modules includes 23 content types and 13 applications. The selection creates a software system targeted at the *presentation* of scientific content. It does not include several collaborative packages. Those packages may be added on demand after installation.

Main Applications

- `ccm-core`: Red Hat Web Application Framework
- `ccm-cms`: Red Hat Content Management System

Content Assets

- `ccm-cms-assets-imagestep: UI step for attaching an image to a content type.`
- `ccm-cms-assets-fileattachment`: Asset to include downloadable files in content items.

- `ccm-cms-assets-notes`: Asset to add further information to a content item.
- `ccm-cms-assets-relatedlink`: Application to add related links to content items.

Content Types

- `ccm-cms-types-address`: Represents an address.
- `ccm-cms-types-article`: The Article content type.
- `ccm-cms-types-bookmark`: Represents a link, usually to an external Web resource.
- `ccm-cms-types-contact`: Encapsulates various bits of information how to contact a person or an organisation.
- `ccm-cms-types-event`: Represents an event including information like location, time frame, etc.
- `ccm-cms-types-filestorageitem`: The File Storage content type for publishing files such as PDF, Word and Excel, enables these items to be reused.
- `ccm-cms-types-formitem`: The Form content type for creating forms.
- `ccm-cms-types-formsectionitem`: The FormSection content type for creating parts of a form.
- `ccm-cms-types-image`: Handles presentation of images, including various metadata.
- `ccm-cms-types-mparticle`: The Multi-Part Article content type for publishing complex articles with multiple sections and images.
- `ccm-cms-types-newsitem`: The News Item content type for publishing news.
- `ccm-cms-types-person`: Represents a person

Applications

- `ccm-cms-publicpersonalprofile`: Creates a basic personal homepage and is able to add automatically preconfigured content.
- `ccm-navigation`: Navigation application for building the navigation structure through published items.
- `ccm-portalworkspace`: Portal application for creating complex customizable front pages and user workspaces.
- `ccm-portalworkspace-homepage`: Extension for ccm-portalworkspace to provide a custom frontpage for a site using a 3 column design.
- `ccm-shortcuts`: The shortcuts applications allows the creation of shortcut links to more complicated URLs.

- `ccm-subsite`: Subsite application is for running multiple websites from one APLAWS server.
- `ccm-ldn-search`: Adds capability to include remote CCM instances into search and combine the results into one integrated list.
- `ccm-ldn-terms`: Creates and maintains domain based categories.
- `ccm-themedirector`: Facilitates management of the look and feel of the website. Allows web designers to upload XSL stylesheets.
- `ccm-ldn-util`: Set of utilities for various tasks.

SCI specific Packages

- `ccm-sci-bundle`: ScientificCMS installation and integration application.
- `ccm-sci-publications: Handles scientific publications in a bibliographically correct way including im- and export.`
- `ccm-sci-personalpublications: Creates automatically a list of publications for a members personal profile (homepage).`
- `ccm-sci-types-member: Represents a member of a scientific institute.`
- `ccm-sci-types-projects: Represents a research project including various science specific metadata.`
- `ccm-sci-personalprojects: Creates automatically a list of projects for a members personal profile (homepage)..`
- `ccm-sci-types-institute: Represents a research institute.`
- `ccm-sci-types-departement: Represents a department of a research institute.`
-

There exist several add-on packages for specific tasks and content types which can be added on demand (cf. below).

3.2 Base Bundle

This bundle contains the minimal set of applications to get an ScientificCMS server operational. It is meant to act as base for a custom set of packages. After installation the `ccm-hostinit` utility program is used to add packages as desired.

Main Applications

- `ccm-core`: Red Hat Web Application Framework
- `ccm-cms`: Red Hat Content Management System

Content Types

- `ccm-cms-types-article`: The Article content type.

Applications

- `ccm-navigation`: Navigation application for building the navigation structure through published items.
- `ccm-portalworkspace`: Portal application for creating complex customizable front pages and user workspaces.
- `ccm-shortcuts`: The shortcuts applications allows the creation of shortcut links to more complicated URLs.
- `ccm-subsite`: Subsite application is for running multiple websites from one APLAWS server.
- `ccm-themedirector: Facilitates management of the look and feel of the website. Allows web designers to upload XSL stylesheets.`
- `ccm-ldn-search`: Adds capability to include remote CCM instances into search and combine the results into one integrated list.
- `ccm-ldn-terms`: Creates and maintains domain based categories.
- `ccm-ldn-util: Set of utilities for various tasks.`
- `ccm-sci-bundle`: ScientificCMS installation and integration application.

Any package of the Standard Bundle not already included may be used as an add-on package and installed on demand to create a custom bundle. Additionally there are dedicated add-on packages described below.

3.3 Demo Bundle

This bundle contains the same set of packages as the Standard Bundle. The default administrator account will be created with the details:

E-Mail: admin@example.com
Forename: Administrator
Surname: Account
Password: 123456
Question: 12345
Answer: 6

It is *not* meant for production use as the name suggests.

3.4 Add-on Packages

Additional Content Types

- `ccm-cms-types-agenda`: The Agenda content type.
- `ccm-cms-types-bookmark`: Creates bookmarks to websites.

- `ccm-cms-types-event`: The Event content type.
- `ccm-cms-types-faqitem`: Stores frequently asked questions and answers.
- `ccm-cms-types-glossaryitem`: Stores glossary terms and definition.
- `ccm-cms-types-htmlform`: The HTML For Content Type.
- `ccm-cms-types-inlinesite`: The Inline site content type for displaying external websites in an iframe.
- `ccm-cms-types-job`: The Job content type for publishing job vacancy announcements.
- `ccm-cms-types-legalnotice`: A Legal Notice for publishing legal notices.
- `ccm-cms-types-minutes`: The Minutes content type for publishing meeting minutes.
- `ccm-cms-types-organization`: Stores organization information.
- `ccm-cms-types-pressrelease`: The Press Release content type for publishing press releases.
- `ccm-cms-types-service`: The Service content type for publishing service information.
- `ccm-cms-types-simpleaddress`: Stores address information in a general format.
- `ccm-cms-types-siteproxy`: Lets you proxy remote sites and style the provided XML.
- `ccm-cms-types-xmlfeed: A content type that lets you make requests to remote sites and style the returned XML.`

Additional Extra Applications

- `ccm-forum`: The forums application is for online discussion groups.
- `ccm-rssfeed`: RSS feed server application.
- `ccm-ldn-exporter`: Exports content objects in XML, suitable for later procesing with the importer application.
- `ccm-ldn-importer`: Application for importing content data from XML files.

More to come soon.

4 Flavours of Distribution

The Software will we delivered in two editions:

- Application Program edition
- WAR File edition

The Application Program edition is easier to install and the preferred installation method if the site doesn't require an integration into an already existing servlet container or standard compliant Web application server.

4.1 Application Program Edition

This distribution is suitable if the installation will not share a servlet container or web application server with other web applications. It combines the application software with a Tomcat servlet container preconfigured to run with minimal installation efforts. It comes as rpm file or `deb` file for Linux based servers.

It can coexist with an already existing servlet container or Web application server. It uses it's own installation locations and support scripts. So even if there is another software application using e.g. Tomcat it may be preferable to use this edition to achieve an improved fault tolerance, if there are no specific integration requirements or server administration considerations.

Proceed as follows:

- Select the appropriate distribution file.
- Install using the operating system's standard procedure
- Log in to the server, switch to the webapps directory and execute a script, which collects required information about database, server name, etc. and creates the registry as well as initially loads the database (CLI interface).
- Optionally install add-on modules from an included repository, and configure it using CLI commands in `~/WEB-INF/bin`.

All bundles are available as an Application Program edition.

4.2 WAR File Edition

This is the preferred distribution method and especially useful if the user already uses a servlet container like Tomcat or Jetty or a Java web application server like jBoss or Glass Fish. The application can friendly live along with other applications[1].

Proceed as follows:

- Select one of the preconfigured war files or create a custom war file.
- Copy the war file onto the server into the desired webapps directory.
- Log in to the server, switch to the webapps directory and execute a script, which collects required information about database, server name, etc. and creates the registry as well as initially loads the database (CLI interface).
- Optionally: Download addon modules, unpack it in the document root directory and install it using the "`ccm load ...`" command in `~/WEB-INF/bin`.

All bundles are available as a war file distribution.

1 Although in its current incarnation an installation as the ROOT application is required. That will change in the near future.

5 Software Pre-Requisites

This chapter outlines the various requirements that must be satisfied before beginning one of the programs distributions can be installed.

5.1 Operating System

The software can be used on any operating system which supports the Java platform. The tested platforms are Linux, Solaris and Windows.

Linux is the preferred operating system. This installation guide will assume the installation is on a RPM based Linux distribution. Red Hat Enterprise Linux provides a product quality platform for running business critical applications. But the software is known to run on other distributions such as Fedora Core, Mandrake and Debian as well. You have to follow the instructions in this guide accordingly. Please check our wiki for further instructions on how to install on these and other distributions.

Microsoft **Windows** is supported as well as Oracle (Sun) **Solaris**. For both operating systems a WAR file distribution has to be used.

Detailed instructions on how to install and configure these pre-requisites on Linux are available in the appendices.

5.2 Java Runtime

A Java runtime environment (JRE) has to be installed on the system. A JRE is sufficient, a JDK, i.e. the Java Software Development Kit, which includes a compiler, is not required.

At least Java version 1.5 is required, version 1.6 is strongly recommended.

Nevertheless, any standards compliant Java machine, regardless of its vendor, should be sufficient.

Oracle (Sun) JRE

The Oracle (Sun) JRE is highly recommended and QA tested. Especially on Linux systems it gives very good performance, scalability and stability.

OpenJDK

OpenJDK is the other QA tested and supported Java machine.

Other Java SDK

There are some other Java implementations around. They should work, too, especially IBMs. But QA testing has be done with Oracle (SUN) and OpenJDK version only.

Specifically the *GCC Java*, which may be delivered on older Linux systems, *doesn't meet* the requirements.

Installation

If a Java JDK is not yet installed by your systems vendor, proceed as follows:

- On **Windows** download the installation kit by Oracle (or any other vendor you prefer) and follow the installation instructions of the vendor.
- On **Linux** you have two installation options. They are described in *Appendix 1*.
- On **Solaris** download Oracle's installation kit and follow the instructions.

5.3 Database

Either an Oracle or PostgreSQL database server is required.

5.3.1 PostgreSQL

The PostgreSQL database software provides a 100% open source solution. It is available for all major **Linux** & **Unix** operating systems and has a port that works on **Windows**.

PostgreSQL 8.x is supported. Previous version 7.x may partly work, but at least some upgrade scripts don't. PostgreSQL *version 9.x is currently not fully compatible,* but should work using version 8.x JDBC driver.

The software is freely available from the projects homepage, www.postgresql.org. On Linux it is preferred to install the packages provided by the distribution if available. *Appendix 2* provides detailed

installation instructions and presents information which may be useful to optimize the distributions configuration, too.

5.3.2 Oracle

The Oracle database software provides an Enterprise grade solution and is available for all tested platforms (Linux, Solaris, Windows).

5.4 Servlet Container

The WAR File Distribution requires a servlet container supporting the Servlet 2.5 and JSP 2.1 specifications. Tomcat 6 is the reference implementation and QA tested, but other vendors products should work as well, if they fulfil the specifications.

The Apache.org Tomcat servlet container provides a 100% open source solution. In combination with Apache it provides support for clustering multiple application servers. You can get Tomcat freely from www.apache.org. For the **Windows** operating system there is a Windows Installer file available. For Linux there exist two installation options, see *Appendix 3* for detailed instructions.

6 Installation Step by Step

This chapter outlines how to install the software and get a server loaded and operational. It focuses on installing the core software packages on Red Hat Enterprise Linux 5 or 6 (or compatibles) or Fedora. The process is common to all bundles. Installing a base distribution for a customized installation is covered in a separate section.

Especially if you install the first time it is advantageous to keep things simple. Therefore you should first perform a *local* installation, i.e. install the Tomcat which is either implicitly installed by the Application Edition or explicitly installed or used by WAR File Edition as a local instance, addressed as `localhost`.

The system is usually run in a multitier, high availability and load balanced environment. In a second step you may adjust the settings to such a more complex production environment. Details are described in the management and deployment guide.

6.1 Preparing the Database Server

As a first installation step a dedicated account must be created on the database server. It is strongly recommended to create a dedicated database user for the content management system to be able to handle security issues. Additionally you must prepare a database into which the installation program can load it's tables and data.

In order to process the following steps you have to login to the database server machine and acquire the necessary privilege (usually *root*).

Creating a PostgreSQL account

- Become UNIX user postgres.

```
[root@localhost root] # su - postgres
[postgres@localhost pgsql] $
```

- Create an unprivileged user account for ScientificCMS e.g. 'ccm', and keeping a record of the password.

```
[postgres@localhost pgsql] $ createuser  -S -R -D -P
ccm
Enter a password for the new role: ccm_xyz
[postgres@localhost pgsql] $
```

  For further information on what these command line options do, run 'createuser –help'

- Create an UNICODE database for the new user, again called e.g. 'ccm'.

```
[postgres@localhost pgsql] $ createdb -E UNICODE -O ccm
ccm
```

- Load the PL/PgSQL stored procedure language into the new database.

```
[postgres@localhost pgsql] $ createlang plpgsql ccm
```

Creating an Oracle account

- Become UNIX user oracle and connect as the sysdba role.

```
[root@localhost root] # su - oracle
[oracle@localhost ] $ sqlplus '/as sysdba'
```

- Create a new user, giving them 200MB of quote on the medium_data tablespace.

```
SQL>  create user ccm identified by ccm_xyz
      default tablespace medium_data
      temporary tablespace temp profile default
      quota unlimited on medium_data;
SQL>  grant connect,resource,query rewrite, ctxapp,
javasyspriv to ccm;
SQL>  revoke unlimited tablespace from ccm;
SQL>  alter user ccm quota 200m on medium_data;
```

6.2 Installing an Application Program Edition

This section covers all bundles distributed as application program file. If you install the Base Bundle you have to follow section 6.5.

Installation of the WAR File edition is covered by the next section.

It is assumed that you decided for the bundle to install and downloaded the appropriate distribution file from a trusted source (e.g. from

http://www.scientificcms.org). You should download the latest spin-off of the software.

- **Step 1**: *Upload* the distribution file to the server machine

  Use an upload program supported by the server. Linux servers may require a ssh based program which require a WINDOWS system to install additional software (e.g. PuTTY, OpenSSH for Windows).

- **Step 2**: *Login* to your machine, become the super user 'root', and change to the directory where you uploaded the rpm file (probably root's home directory).

- **Step 3**: Execute the RPM program to install the bundle. In case of the standard bundle perform:

```
[root@localhost ROOT]# rpm -i scientificcms-2.0.0-
std.rxxxx.noarch.rpm
```

- **Step 4**: *Switch user account* temporarily to scicms and make ~/WEB-INF/bin as default

```
[root@localhost ROOT]# su -
-bash-4.1$ pwd
/usr/share/scientificcms
-bash-4.1$ cd   webapps/ROOT/WEB-INF/bin
-bash-4.1$
```

  This step is **most important**! If you miss it the next step will fail!

- **Step 5**: *Prepare* the system for operation

  Execute within the ~/WEB-INF/bin directory:

```
-bash-4.1$  ./ccm prepare
```

  This is the most important and delicate step! The previous invoked command displays a menu where all required information has to be entered. Consult section 6.4 in case of questions and return to finish installation.

  After finished all preparation steps leave scicms account and return to root:

```
-bash-4.1$  exit
[root@localhost ROOT]#
```

- **Step 6**: *Start ScientificCMS*

```
[root@localhost ROOT]# service scientificcms start
```

Wait for the initialization process to complete. This typically takes 30-60 seconds.

- **Step 7**: *Check* if everything works OK.

The easiest way is to point web browser to localhost on port 8090.

```
root@localhost# links http://localhost:8090/
```

Logs recording initialization process and any errors that may have occurred can be found in /var/log/scientificcms.

If you inspect the log file you may find several warnings about existing primary keys which can safely be ignored.

You will also find error messages about manager and host-manager. Both require configuration before they can be used.

If ScientificCMS fails to start, check the port configuration. In order to avoid conflicts with an already installed servlet container or web application server ScientificCMS uses different ports for operation as the usual server applications.

Protocol	Standard Port	Assignment	ScientificCMS Port	Assignment
Shutdown	8005	(free)	8015	(free)
Ajp 1.3	8009	ajp13	8019	(free)
HTTP	8080	Http alternate	8090	HTTP alternate
HTTPS	8443	Apache SSL	8453	(free)

You can modify the ports in /etc/scientificcms/server.xml. You have to track all port modifications in ~/WEB-INF/conf/ccm-core/web.properties.

If everything works as expected proceed with the post-installation procedures.

6.3 Installing a WAR File Edition

This section covers the standard, extended, and demo bundle distributed as war file. RPM distribution installation is covered by the previous section. And a separate section covers the installation of a base bundle.

It is assumed that you decided for the bundle to install and downloaded the appropriate WAR distribution (e.g.). You should download the latest spin-off of the software.

- **Step 1**: *Upload* the software to the server machine

  Use an upload program supported by the server. Linux servers may require a ssh based program which require a WINDOWS system to install additional software (e.g. PuTTY, OpenSSH for Windows).

- **Step 2**: *Login* to your machine and become the super user 'root'. Change directory to the webapps directory of Tomcat or your installed servlet container.

  Ensure that you have read / write access to the directory and all subdirectories.

- **Step 3**: *Create directory* ROOT if it not already exists and change to this directory as default.

```
[root@localhost webapps]# cd ROOT
[root@localhost ROOT]#
```

  If there exists a directory ROOT ensure that it is empty.

```
[root@localhost ROOT]# ls -al
[root@localhost ROOT]# rm -rf *
```

- **Step 4**: *Unzip* the war file into the (default) ROOT directory.

```
[root@localhost ROOT]# unzip /home/[thisuser]/aplaws-2-
0-x-rrr.war
```

- **Step 5**: Make *~/WEB-INF/bin as default*:

```
[root@localhost ROOT]# cd WEB-INF/bin
[root@localhost bin] #
```

- **Step 6:** The installation environment may be not able to perserve file permissions under Unix / Linux. So we have to adjust them:

```
[root@localhost bin]# chmod ug+x  ccm*
[root@localhost bin]# chmod ug+x  libexec/ant/bin/an*
[root@localhost bin] #
```

- **Step 7**: Execute the ccm tool:

```
[root@localhost bin]$ ./ccm prepare
```

If an error message appears like './ccm no permissions' try "`sh ccm prepare`" instead.

The command opens a configuration menu where you have to enter the required information. For detailed instructions jump to chapter 6.4 and return when finished to follow the next steps.

- **Step 8**: Check and *adjust permissions*

The user who owns the tomcat process must have read permissions in the complete ROOT directory tree and write access for the subdirectories "html", "WEB-INF/work", "WEB-INF/conf", "themes/published-themedir", and "templates". From the ROOT directory execute:

```
[root@localhost ROOT]# chown -R tomcat  html
[root@localhost ROOT]# chown -R tomcat  WEB-INF/work
[root@localhost ROOT]# chown -R tomcat  WEB-INF/conf
[root@localhost ROOT]# chown -R tomcat  templates
[root@localhost ROOT]# chown -R tomcat
themes/published-themedir
```

Replace "tomcat" by the actual user who owns the tomcat process.

If you use tomcat as your front end web server to actually deliver content (instead of hiding behind an Apache server – see maintenance manual) is is preferable to make as less as possible directories writeable for the process.

If other persons as the server's system administrator should be able to alter configuration parameters or to create new themes you should create an appropriate group, e.g. ccm-admin, and grant write access:

```
[root@localhost ROOT]# groupadd ccm-admin
[root@localhost ROOT]# groupmems -a [thisuser] -g ccm-
admin
[root@localhost ROOT]# chgrp -R ccm-admin *
[root@localhost ROOT]# chgrp ccm-admin .
[root@localhost ROOT]# chmod -R g+w *
[root@localhost ROOT]# chmod g+w .
```

Be carefull with these commands and control thoroughly for any typo! Repeat the groupmems -a command for all members who should maintain the instance.

- **Step 9:** Optionally adjust Tomcat Installation

  You might adjust the Tomcat runtime environment to optimize memory allocation. requires at least 512 mb, preferable at least 768 or even 1024 mb.

  On a Red Hat (or compatible) system and most other rpm based distributions you will find a file //etc/tomcat6/tomcat6.conf file where is the place to make adjustments.

  Look for a line

```
JAVA_OPTS="-Xminf0.1 -Xmaxf0.3"
```

  and replace it with

```
JAVA_OPTS=”-Xmx768m -Xms512m -XX:MaxPermSize=256m"
```

  You may choose 1024 or even higher for Xmx depending on your systems memory and usage.

- **Step 10**: Start Tomcat / your servlet container.

  On rpm based systems execute:

```
[root@localhost ROOT]# service tomcat6 start
```

  On other systems use the appropriate command instead.

  Wait for the initialization process to complete. This typically takes 30-60 seconds.

- **Step 11**: Check if everything works OK.

  The easiest way is to point web browser to localhost on port 8080.

```
root@localhost# links http://localhost:8080/
```

If you had to use another port number replace it appropriately.

Logs recording initialization process and any errors that may have occurred can be found in /var/log/ccm on a Linux system.

If everything works as expected proceed with the post-installation procedures.

6.4 Prepare the System for Operation

This is the most important and delicate step! It will prepare the database for usage by (create all the tables and write some basic information into them) and write the basic information for database access into configuration files. If you execute the command "`sh ccm prepare`" as explained in the step-by-step instructions, the system prompts for the required information using a simple menu system:

```
Starting CCM-Tool...

====================================
prepare:
- Initializes ccm after installation.
- Execute only ONCE!
====================================

  [1] Set JDBC connection URL: null
  [2] Set Server virtual host: null
  [3] Set Administrator Email Address: null
  [4] Set Administrator First Name: null
  [5] Set Administrator Last Name: null
  [6] Set Administrator Password: null
  [7] Set Administrator Password Question: null
  [8] Set Administrator Password Answer: null

  [l] List parameters  [o] Show optional parameters
  [v] Validate parameters
  [e] Exit  [r] Set required parameters  [A] Abort  [?]
Help

Choose:
```

To provide the requested information type the character in front of a requested parameter, e.g. "`1 [return]`"for setting JDBC connection

```
Choose: 1 <ret>
JDBC connection URL:
```

For database connection you have to provide the information as configured in chapter 6.1:

- On PostgreSQL:

```
jdbc:postgresql://localhost/ccm_db?
user=ccm_usr&password=ccm_xyz
```

- On Oracle:

```
jdbc:oracle:oci8:ccm/ccm_xyz@ora9i
```

Replace the host part (localhost), database name (ccm_db), user name (ccm_usr) and password (ccm_xyz) appropriately. After typing in the URL the menu shows the prompt "`Choose:`" again.

Walk through the menu and provide all requested information.

The host name is typically the machines primary internal name or localhost. If you use a virtual host whose "symbolic" name points to your real machine (e.g. `www.myhost.org` as an alias for server01.mybusiness.com) you must enter that symbolic name. If you install the Application Edition the port number to append is 8090. If you install a WAR File Edition into an existing or specifically installed servlet container or Web Application Server you have to use the port number configured. A typical 'Server Virtual Host' option value will be:

```
localhost:8090
```

If you are using Tomcat and it is already running to serve other web services, those instance will likely use 8080! If you operate the servlet container behind an apache (e.g. using the ajp protocol) you must enter the name known to apache and the apache port, usually 80.

While typing in the information the List-Key [`l`] will update the list displayed at the beginning with the actually given information. The optional parameter list [`o`] is a toggle and hides or additionally displays all optional parameter. **Do not** use this key **before** you provides the administrators email, first, and last name! The Required Parameter key [`r`] is a convenience way to prompt for all required parameters without the need to select each individually. The Verify-Key [`v`] checks all parameters and should be used before exiting the menu.

In the end it will look something like:

```
Choose: v
  -- valid --
Choose: l

  [1] Set JDBC connection URL:
          jdbc:postgresql://localhost/ccm-ldnstd?
user=ccm&password=ccm
  [2] Set Server virtual host: localhost:8080
  [3] Set Administrator Email Address: webmaster@.org
  [4] Set Administrator First Name:
  [5] Set Administrator Last Name: Admin
  [6] Set Administrator Password: 123456
  [7] Set Administrator Password Question: 12345
  [8] Set Administrator Password Answer: 6789

  [l] List parameters  [o] Show optional parameters
  [v] Validate parameters
  [e] Exit  [r] Set required parameters  [A] Abort  [?]
Help

Choose: e
```

The [e] leaves the menu, saves all entries in the system's configuration registry and starts loading the database. It produces a lot of progress messages and some warnings, e.g. “table already has a primary key”, which can safely be ignored.

Since it will take some time, progress messages will be output to the console when running this command. When it gets to load the category systems expect about a 10 minutes wait. Rather large and complex category systems will have even longer load times, up to about 40 minutes.

When the process has finished return to the next installation step.

6.5 Adding Packages to an Installed Bundle

In some cases it might be necessary to customize an installed bundle according to specific requirements of a site, either by adding a single package (or even a set of packages) provided by the project or by a third-party developer.

6.5.1 Prerequisites

The project optionally provides an addon pack for the various bundles which include packages not included in a bundle. Download the appropriate bundle and store it on the server which is hosting .

- **WAR File Edition**:

  The Add-on pack is provided as a zip file. Expand the zip file at a location where it can be accessed from within the directory tree.

  The home directory of the administrator is a good place. Make a note where you stored the pack (e.g. /home/my-admin/addons)

- **Application Program Edition**:

  The Add-on packs are provided as rpm file. Store the file at a location on the server where you can install using the rpm tool. Execute the RPM installation process. The data are stored inside the directory at /srv/ so they are accessible at any time.

  RPM installation requires root privileges. Package installation requires just admin privileges for the application tree (my-admin as suggested previously). Alternatively the War File Edition can be used by a administrator without depending on a system administrator with root privileges. The content of the packs is the same, just how to invoke the installation program differs slightly.

6.5.2 Installation

Ensure to perform the installation of add-on packages not before having completely finished the installation process, including the first start of the system! Otherwise the process will fail!

War File Edition

1. Login to the server and change to ~/WEB-INF/bin directory.
2. Check, which modules are available

```
[@localhost bin]# CCM_REPO=/home/my-admin/addons sh \
> ccm-hostinit  list
```

   A list of available modules will be displayed.

3. If you need information about one or more modules execute

```
[@localhost bin]# CCM_REPO=/home/my-admin/addons sh \
> ccm-hostinit  info ccm-portalserver ccm-faq ccm-
docrepo
```

   The system displays a short description of each package.

4. Deploy selected modules into the application tree. It is not necessary to stop the servlet container at this time. But because the web.xml configuration file will likely be modified the servlet container, if running, will try to reload (that is the default configuration of most servers). This will fail unless the next step is finished.

```
[@localhost bin]# CCM_REPO=/home/my-admin/addons sh \
> ccm-hostinit  add ccm-portalserver ccm-faq ccm-docrepo
```

   Ensure that you enter a space delimited list of packages. Avoid any typo.

   The system displays a lot of information, the last message with “build successful”.

5. Load the base data of the deployed packages into database and configuration registry:

```
[@localhost bin]# CCM_REPO=/home/my-admin/addons sh \
> ccm  load ccm-portalserver ccm-faq ccm-docrepo
```

   Ensure that you enter a space delimited list of identical packages as in the previous step!. Avoid any typo.

6. Restart the server either by restarting the servlet container or by initializing a reload of :

```
[@localhost bin]# touch ../web.xml
```

   After a short period of time the servlet container should process a reload.

7. Check whether the system is operational using a browser or by inspecting the log file.

8. Finished.

War File Edition

1. Login to the server and change to ~/WEB-INF/bin directory.

2. Check, which modules are available

```
[my-admin@localhost bin]# sh ccm-hostinit  list
```

   A list of available modules will be displayed.

3. If you need information about one or more modules execute

```
[my-admin@localhost bin]# sh ccm-hostinit  info ccm-
portalserver ccm-faq
```

   The system displays a short description of each package.

4. Deploy selected modules into the application tree. It is not necessary to stop the servlet container at this time. But because the web.xml configuration file will likely be modified the servlet container, if running, will try to reload (that is the default configuration of most servers). This will fail unless the next step is finished.

```
[my-admin@localhost bin]# sh ccm-hostinit  add \
ccm-portalserver ccm-faq
```

   Ensure that you enter a space delimited list of packages. Avoid any typo.

   The system displays a lot of information, the last message with “build successful”.

5. Load the base data of the deployed packages into database and configuration registry:

```
[my-admin@localhost bin]# sh ccm  load \
ccm-portalserver ccm-faq
```

   Ensure that you enter a space delimited list of identical packages as in the previous step!. Avoid any typo.

6. Restart either by restarting the servlet container or by initializing a reload of :

```
[my-admin@localhost bin]# touch ../web.xml
```

   After a short period of time the servlet container should process a reload.

7. Check whether the system is operational using a browser or by inspecting the log file.

8. Finished.

7. Post-install procedures

After installation you should consult the **Maintenance Guide** for further actions to take. Some quick hints follow here.

7.1 Initial Runtime customisation

Immediately after installation you should perform some basic adjustments.

- Since the public facing **name of the site** will typically be different from the internal host names, the waf.server parameter should be changed to reflect this. There are 3 parameters:
  - waf.web.host: Sets the hostname and port of the machine on which the CCM instance is running
  - waf.web.server: Sets the hostname and port that users of a site will see in URLs generated by WAF
  - waf.web.site_name: The name of your website, for use in page footers for example

  Use

  ```
  [root@localhost bin]# ccm set waf.web.hostg=www.mysite
  ```

  to set the site name.
- Another adjustment is the **login name**. By default a user's email address is used. You may switch to a freely selectable user name by executing

  ```
  [root@localhost bin]# ccm set
  waf.kernel.primary_user_identifier=screen_name
  ```
- There are a couple of parameters that should be tuned for optimal **performance**. This is done using the 'ccm set' command while ~/WEB-INF/bin is the default directory. The most important one is jdbc_pool_size.

  ```
  [root@localhost bin]# ccm set
  waf.runtime.jdbc_pool_size=100
  ```

In order to activate the new settings you have to restart the web application.

All the parameters you specify with the ccm tool will be stored in the local file system (`~/WEB-INF/conf/registry`).

7.2 Content Administration

After having completed the installation you may point your browser to localhost:8080 and the portal home page is shown. In order to add content, maintain user and workflow, there are several addresses available:

Table 7.1.: Administration Pages

URL	Description
/ccm/content-center	The content centre where you will manage the content of the site
/ccm/ds/	Web developer support
/ccm/admin/terms/	Terms admin
/ccm/atoz/admin/	A-Z admin
/ccm/search/admin/	Search admin
/ccm/register/change-password	Change your password
/ccm/register/edit-profile/	Edit profile
/ccm/admin/auth-http/	HTTP authentication admin
/ccm/register/logout	Log out
/ccm/portal/	The site portal
/ccm/admin/shortcuts/	Shortcuts admin
/ccm/admin/subsite/	Subsite Admin
/ccm/admin/themes/	Themes admin
/ccm/admin/	User and group administration

Check the administration guides for further information.

Part II
Update

This part describes how to update an existing ScientificCMS installation and the necessary precautions of avoid any lost of data and configuration.

For version 2 of ScientificCMS there is currently no update applicable.

Part III
System Maintenance

This part introduces various tasks of maintenance and deployment to ensure long-term hassle-free operations of ScientificCMS.

Specifically it describes various ways of integrating ScientificCMS into an Apache environment, the distribution of individual components on different servers, performance tuning, and other deployment issues.

8 Web Server Integration

8.1 Overview

There are several reasons for integrating a ScientificCMS servlet container with a Web Server as the Apache, instead of using its build in HTTP server. Servlet Container like Tomcat, Jetty or Resin are fast nowadays, but Apache – or any other WEB server – is optimized for handling http requests and has a lot of additional features.

Integrating with a Web Server is mandatory, if you have to serve another web site on the same machine using a universal web server which can handle e.g. PHP. Using a servlet container directly for deploying ScientificCMS would require to use a non-standard http port as 8080 (default for Tomcat). Most user don‘t know about that port and may never find your site, especially if they are protected by a very restrictive firewall.

So it is a common scenario in a productive environment to use a standard web server like Apache to act as a front end for your ScientificCMS servlet container. There are four methods to do so:

- use of redirection
- use of reverse proxying
- use of a specialised proxy technique based on JK protocol.
- use of specialised connectors using the JK protocol

These configuration options are in no way specific to ScientificCMS but are standard administrative tasks and require standard administrator's skills. For your convenience this guide will provide step by step instructions for each of these methods and discuss their pros and cons.

8.2 Preliminary Step: Setting Up the Web Server

- Ensure, the web server is up and running properly. Open a Web browser and point it to the address of your server. You should see either the default web page of your installation or the previously prepared content for the site. If you are running Red Hat

Enterprise Linux and did a fresh installation, the default web page is stored in `/var/www/html`.

- Next you have to decide weather you will use the default server address for your ScientificCMS or install a dedicated server instance (virtual server). In this case you must configure the virtual server now and ensure that it is working properly.

The following steps depend on a properly working web server configuration.

8.3 Alternative A: Redirection

Redirection is a method to inform the visitors web browser, that the intended content can be found at another location so that the browser can automatically visit the other address. All current browser programs support this feature. For browsers which don't you have to provide a link.

The pro of this method is its minimal configuration effort and it works for all Web servers. No special capabilities are needed. If you don't have access to the web server configuration it may be your only option.

You have just to prepare a html file containing the appropriate instructions. You can put this file in your Web servers root as default page so that every request is automatically redirected. You can also put it into a subdirectory so that only a special address is redirected and that way you can mix ScientificCMS provided pages with standard Web pages.

The cons is, that it is still the servlet container which serves the clients requests and the client has to use port 8080 to access the pages. It is only the first contact which is served by the web server, all other request are directly served by the servlet container, just as the standard installation does. So one can argue it is not an integration at all.

If you use the address `www.mysite.org` you just have to prepare the following html file:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>www.mysite.org</title>
  <meta name="GENERATOR" content="Quanta Plus" />
  <meta http-equiv="Content-Type"
          content="text/html; charset=iso-859-1" />
  <meta http-equiv="REFRESH" content="0;
              URL=http://www.mysite.org">
  <style type="text/css">
  /* <![CDATA[ */
#messagebox {
        position:absolute;
        top:180px;
        left:0px;
        width:100%;
        margin:0px 0px 0px 0px;
        padding:0px 0px 0px 0px;
        border-style: groove;
        border-color:black;
        border-width:0px 0px;
        horizontal-align:middle;
        text-align:right;
        }
 /* ]]> */
 </style>
</head>

<body>
 <div id="messagebox">
   <p style="text-align:center;" >
    Unfortunately, your browser doesn't support
automatic redirection.
    Please click the provided link:
   </p>
   <p style="text-align:center;" >
    <a href="http://www.mysite.org:8080">
       http://www.mysite.org:8080</a>
   </p>
 </div>
</body>

</html>
```

Please, replace the servers address as appropriate.

8.4 Alternative B: Proxy

Using the reverse proxy method the web server requests the pages from the ScientificCMS servlet container in behalf of its client and passes the retrieved pages to the client in behalf of the servlet container. This process is perfectly hidden from the client. There is no

need for the client to use the servlet containers port 8080 which may cause difficulties with some firewall configurations. This method is very flexible. The servlet container can reside on the same machine or on another one, even behind a firewall. You don‘t have to modify the configuration of the servlet container, but you may alter its configuration to accept connection from your web server only in order to increase security. The only requirement is the web servers ability to act as a reverse proxy. The Apache server can do so, the configuration is quite easy. Because the communication between the web server and the servlet container uses a standard TCP/IP connection as well as the HTML protocol and the servlet container still have to work as a web server, too, there is no performance gain using this integration method.

8.4.1 Configuring the Apache Server

We will use the Apache server as an example to explain the web server configuration. Please, consult your web servers manual to perform the required steps accordingly. First you must ensure that the proxy module is loaded. If you use Red Hat Enterprise Linux version 5 or later loading the module is pre-configured. Inspecting the Apache configuration file (usually `/etc/httpd/conf/httpd.conf`) you will find a bunch of LoadModule instructions, look for the following lines:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_connect_module
modules/mod_proxy_connect.so
```

If it isn‘t there inspect the modules directory for existence and insert the lines at the appropriate place, along with the loading of other modules. If the module isn‘t part of your distribution you have to recompile the Apache. If you are using Apache version 1.x walk through the file looking for a block of „AddModule“ instructions. Add lines for the proxy modules if they aren‘t already there. Apache version 2.x performs the AddModule step automatically. If you use virtual hosts you must put the following directives into the virtual host container. Otherwise you should place it near the end of section 2 of the Apache configuration file:

```
<IfModule  mod_proxy.c>
   ProxyRequests Off
   <Proxy *>
      Order deny,allow
      Allow from all
   </Proxy>

   ProxyPreserveHost  On
   ProxyPass           /       httpd://localhost:8080/
   ProxyPassReverse    /       httpd://localhost:8080/
</IfModule>
```

The ProxyPreserveHost directive is mandatory if you place the instructions into a virtual host container. Otherwise it works like a simple redirect. You may omit it if you place the instructions into section 2 of the configuration file.

You can improve performance by activating caching. Add the following directives immediately beyond the ProxyPassReverse directive and before the closing </IfModule> directive:

```
<IfModule mod_disk_cache.c>
   CacheEnable  disk  /
   CacheRoot    „/var/cache/mod_proxy“
</IfModule>
```

In this configuration the complete document root of the apache server is fetched from the corresponding directory of the APLAWS server. Proxying from document root is mandatory, because otherwise the URLs to the various APLAWS resources don‘t match its real locations and the page will be broken. You can‘t use any of the Apache directory directives, e.g. authorization requests. You even don‘t need a document root directory.

8.5 Alternative C: Proxy Using a Binary Protocol Extension

Since *version 2.1* the Apache Web server provides an extension for the proxy module described above which uses a specialised communication protocol between Apache and the Tomcat servlet container. It is an adaptation of the development of *connectors* and the *Apache JServ Protocol* (AJP) as described in the next section. It uses the proxy method but the AJP protocol for communication instead of HTTP.

Because of the proxy methodology is has basically the same characteristics as alternative B described above. The Tomcat servlet container is

perfectly hidden from the client. It can reside on the same machine or on another one, even behind a firewall, etc.

Because of the binary module this technique is bound to the Apache server as long as other server programs don't provide the protocol extension. Same is true to the servlet container side, it must support the AJP protocol. Tomcat does.

Because the communication between the web server and the servlet container uses an optimized protocol you will achieve some performance gain using this integration method.

This technique is the recommended way to integrate a servlet container, it is easy to configure and efficient to use. For further details see: http://httpd.apache.org/docs/2.4/mod/mod_proxy_ajp.html

8.5.1 Preparing the Servlet Container

The servlet container must support the AJP protocol. In its standard configuration Tomcat supports it by default, and ScientificCMS makes no difference. Check the file server.xml in either `/etc/scientificcms` or in the tomcat configuration directory used by your installation (usually `/etc/tomcat6` or `/usr/share/tomcat6/config`, replace the version number appropriately).

In this file you will find 2 lines as

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" protocol="AJP/1.3"
redirectPort="8443" />
```

directly after a section "`<Connector port="8080" protocol="HTTP/1.1 ...`". If it isn't there add it appropriately.

Note the port number, you need it in the next step. 8009 is the default value. It must be unique, so if you run multiple Tomcat servers on your machine, you must configure each to use a different port! Refer to section XX about using multiple Tomcat instances.

8.5.2 Configuring the Apache Server

The Apache configuration is quite easy. First check its version, 2.1 or above is required.

Step 1: Check Configuration

Because this technique uses a protocol extension, you must ensure that the proxy module `mod_proxy` as well as the extension module `mod_proxy_ajp` is loaded. If you use Red Hat Enterprise Linux version 5 or later loading the module is pre-configured. Inspecting the Apache configuration file (usually `/etc/httpd/conf/httpd.conf`) you will find a bunch of LoadModule instructions, look for the following lines:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

Important are the first and the last line. If they aren‘t there inspect the modules directory for existence and insert the lines at the appropriate place, along with the loading of other modules. If the module aren‘t part of your distribution you have to recompile the Apache.

Step 2: Activate Proxy

To activate the proxy you have to put the following directives into the virtual host container, if you use virtual hosts, or to place it near the end of section 2 of the Apache configuration file:

```
<VirtualHost *:80>
    ServerName www.myserver.org
 #   DocumentRoot    /srv/myserver/html
    DirectoryIndex  index.jsp

   <IfModule  mod_proxy_ajp.c>
      ProxyPass         /       ajp://localhost:8009/
   </IfModule>

   ServerAdmin     webmaster@myserver.org
   ErrorLog        logs/proaktiv-error_log
   CustomLog       logs/proaktiv-access_log combined
</VirtualHost>
```

If the URL path on the proxy differs from that on the servlet container, a `ProxyPassRevers` directive may be required. Add beyond the `ProxyPass` directive something like

```
ProxyPassReverse /apps/ccm http://www.myserver.org/ccm
```

But such a configuration may add a lot of trouble so you are strongly advised to deploy the application on the Tomcat server at the same path as the proxy.

Sometimes you may want to directly access a directory in the filesystem, e.g. to deliver large files as videos without storing them in the database. In order to accomplish that you must specify a document root, create a subdirectory (or several subdirectories) to store the files, and instruct mod_proxy not redirect that directory (or directories).

As an example, if you wish to deliver files from directories `/video` and `/downloads`, your configuration file would be something like

```
<VirtualHost *:80>
    ServerName www.myserver.org
    DocumentRoot    /srv/myserver/html
    DirectoryIndex  index.jsp

   <IfModule  mod_proxy_ajp.c>
      ProxyPass         /video !
      ProxyPass         /downloads !
      ProxyPass         /       ajp://localhost:8009/
   </IfModule>

   ServerAdmin     webmaster@myserver.org
   ErrorLog        logs/proaktiv-error_log
   CustomLog       logs/proaktiv-access_log combined
</VirtualHost>
```

8.5.3 Security

Despite the Apache integration ScientificCMS is still accessible via port 8080 or 8090 at the current state. You have two options to prevent access:

- configure a firewall to block port 8080 /8090
- reconfigure scientificcms / tomcat not to listen on port 8080 / 8090 for public access

Configuring a firewall is the easier path and doesn‘t bear the risk of messing up with the ScientificCMS configuration or conflict with future updates. In RHEL use system-config-securitylevel to activate the build in firewall. Double check to allow http/https request as well as ssh (or telnet) to enable remote administrator access. There is also a command line interface available: system-config-securitylevel-tui. For other distribution consult the manual.

8.5.4 Troubleshooting

- In case of any difficulties you should use `netstat -lp | grep 8009` (modify according to the port number you configured) to determ-

ine weather the port is used and which process is listening on the it.

- If Tomcat is running on a different server, check the name or address in the `ProxyPass` directive.
- If Tomcat is running on a different server, check the firfewall configuration for port 8009 (or the port you actually use).
- If there is an error message ala “permission denied” and selinux is active, temporarily disable selinux by issuing `setenforce 0` and check again. It it works, adjust selinux. Don't forget to re-activate selinux by `setenforce 1`.

8.6 Alternative D: Integration Using Connectors

Using this method is logically very similiar to the proxy method, but uses a specialized dedicated communications protocol for efficiency and performance. The pros and cons are nearly the same as for the proxy method with the exception that the configuration requires a lot more of effort. And your servlet container must provide a connector which is compatible with your Web server. Apache Tomcat provides connectors for Apache (mod_jk), MS IIS (isapi redirector) and Netscape / Sun One (nsapi redirector). For detailed information you may have a look at the tomcat project site[2]. As an example we will describe the Apache integration.

Installation requires the following steps:

- The module mod_jk[3] isn‘t part of of RHEL (nor most other distributions). The Apache Tomcat project provides binaries for the current Apache version. For older versions you may find binaries at jpackage.org. Otherwise you have to download and compile the sources.
- Modification of the Apache httpd.conf file to load the module and perform its Apache side configuration.

2 See http://tomcat.apache.org/connectors-doc/

3 You may find references to mod_jk2 in the literature, which was supposed to be a successor of mod_jk. But its development has been canceled in 2004 and it is unsupported now.

- Creating a workers.properties file to configure the Tomcat side of its configuration. Good news are that Tomcat version 5.5 and above is already prepared to work with mod_jk by default.h

Although the Apache version 2.2 and beyond comes with a specialized proxy module for connecting to a tomcat, which uses a lot of mod_jk code, using connectors has still advantages in terms of configuration options and features and is preferred by many Tomcat developers.

8.6.1 Installing mod_jk

Copy the downloaded or compiled module into the Apaches modules directory (usually `/usr/lib/httpd/modules`). Rename it to `mod_jk.so` and make it executable if necessary.

8.6.2 Modifying the httpd.conf file

If you are using RHEL or a compatible additional modules are installed and configured by placing a configuration file into the directory `/etc/httpd/conf.d`. There is no need to manipulate the main httpd.conf file. Open an editor of your choice creating a file „`mod_jk.conf`“ in that directory:

```
LoadModule      jk_module       modules/mod_jk.so

JkWorkersFile   /etc/httpd/conf/workers.properties
JkLogFile       logs/mod_jk.log
JkLogLevel      error

##################################################
#
#   SSL configuration skipped
#   usually not needed for aplaws
#
##################################################

# Root context mounts for Tomcat
#
# The complete document root directory must be
# mounted. Otherwise the paths for content,
# graphics, etc don't match.
#
# Should normally be defined in a virtual host
# container. Uncomment here it the apache does
# nothing else as working as a front end for an
# aplaws servlet container.
#
# JkMount  /*  <replaceWithYourInstance>
```

8.6.3 Creating a workers.properties file

Using your favourite editor create a file workers.properties in `/etc/httpd/conf` and add the following content:

```
# workers.properties
#
# configuration for the apache mod_jk plugin
#

# list of workers the plugins should create
worker.list=ccm

#
# worker definitions
worker.ccm.port=8009
worker.ccm.host=localhost
worker.ccm.type=ajp13
```

Port 8009 is the default Tomcat configuration for AJP protocol.

8.6.4 Creating the JK Mount Instructions

Next you have to create the mount instruction for the appropriate virtual server container. If you use virtual servers the instructions must be inserted into the appropriate directives block (some distributions use separate config files for each virtual server, usually in `/etc/httpd/vhost` or alike). Otherwise the must be included near the end of the main config file.

There are two alternatives: If the Apache should work as a front end only and should not serve static page contents or perform authorization, include the following lines:

```
<VirtualHost *:80>
       ServerName      www.myserver.org
       DirectoryIndex  index.jsp
       <IfModule  mod_jk.c>
               #forward all requests to Tomcat
               JkMount         /*  ccm
       </IfModule>
       ServerAdmin  aplaws-admin@myauthority.gov.uk
       ErrorLog     logs/myserver.org-error_log
       CustomLog    logs/myserver.org-access_log common
</VirtualHost>
```

This setup is pretty simple, you don‘t even need a document root. Because of that you can‘t use any of the Apache‘s security features. As an example, a visitor, who knows the internal APLAWS+ directory structure, can request listings of the directories in question, navigate

through the directory tree and inspect the files content. But this is not specific for Apache integration but is true for the standard APLAWS+ installation using Tomcat as well.

If you would like to use Apache for basic authorization or for serving other static content you must use a more complex setup:

```
<VirtualHost *:80>
    ServerName      www.myserver.org
    DocumentRoot    /www/myserver/web
    DirectoryIndex  index.jsp  index.html
    <Directory /www/myserver/web>
            AuthType        Basic
            AuthName        "myauthority"
            AuthUserFile    /www/myserver/.htusers
            Require valid-user
    </Directory>
    <IfModule  mod_jk.c>
            #forward aplaws requests to Tomcat
            JkMount         /*.jsp  ccm
            JkMount         /ccm/*  ccm
            JkMount         /__c*   ccm
            JkMount         /ass*   ccm
            JkMount         /jav*   ccm
            JkMount         /ccm*   ccm
            JkMount         /css*   ccm
            JkMount         /con*   ccm
            JkMount         /hel*   ccm
            JkMount         /red*   ccm
    </IfModule>
    ServerAdmin     admin@myserver.org
    ErrorLog        logs/myserver.org-error_log
    CustomLog       logs/myserver.org-access_log common
</VirtualHost>
```

With this setup you can use Apaches authorisation mechanism (otherwise omit the directory container), protect the APLAWS+ installation against directory listings and individual file access, and may let Apache serve static content in other directories beyond the document root. You may have to add more mount points using JkMount if your APLAWS+ configurations uses additional directories. Inspect the log files to identify missing contents and its location.

Depending on your needs a slightly different configuration may be easier to maintain:

```
<VirtualHost *:80>
    ServerName      www.myserver.org
    DocumentRoot    /www/myserver/web
    DirectoryIndex  index.jsp  index.html
    <Directory /www/myserver/web>
            AuthType        Basic
            AuthName        "myauthority"
            AuthUserFile    /www/myserver/.htusers
            Require valid-user
    </Directory>
    <IfModule  mod_jk.c>
               #forward all requests to Tomcat
               JkMount          /*  ccm
               #but don't forward
               JkUnMount        /mylocal_1  ccm
               JkUnMount        /mylocal_2  ccm
               JkUnMount        /mylocal_3  ccm
    </IfModule>
    ServerAdmin     admin@myserver.org
    ErrorLog        logs/myserver.org-error_log
    CustomLog       logs/myserver.org-access_log common
</VirtualHost>
```

This way you forward all request to Tomcat but those specified in the JKUnMount directive. This variant is easier to maintain if you have a limeted amount of static directories to serve an if these directories follow a simple naming pattern.

8.6.5 Test the configuration

You are done. Restart the Apache and test. If an error occurs inspect the log files in `/var/log/httpd` to identify the cause. You should check that postgresql as well as ccm and Apache are automatically started at boot time. Ensure that ccm starts before Apache.

8.6.6 Protecting the servlet container

Despite the Apache integration aplaws is still accessible via port 8080 at the current state. You have two options to prevent access:

- configure a firewall to block port 8080
- reconfigure ccm / tomcat not to listen on port 8080 for public access

Configuring a firewall is the easier path and doesn‘t bear the risk of messing up with the APLAWS+ configuration or conflict with future updates. In RHEL use system-config-securitylevel to activate the build in firewall. Double check to allow http/https request as well as ssh (or

telnet) to enable remote administrator access. There is also a command line interface available: system-config-securitylevel-tui. For other distribution consult the manual.

8.6.7 Troubleshooting

- In case of any difficulties you should use `netstat -lp | grep 8009` (modify according to the port number you configured) to determine weather the port is used and which process is listening on the it.
- You have set up mod_jk closely following the instructions and neverthless get the message: “Service temporarily not available”. In ~/logs/mod_jk.log you find

```
ajp_connect_to_endpoint::jk_ajp_common.c (877): Failed
connecting to tomcat. Tomcat is probably not started or
is listening on the wrong host/port (127.0.0.1:8009).
Failed errno = 13
```

- If you use a recent Linux distribution you should check, if selinux is enabled (use `less /etc/selinux/config`). If it is is you may disable is temporarily (use `echo 0 > /selinux/enforce` as root, use 1 to re-enable) to check.
- To permanently disable selinux, use the tool provided by the distritution or set SELINUX=disabled in /etc/selinux/config.

9 Performance Tuning

ScientificCMS is quite a complex system and needs some amount of computing resources to work in a timely manner. If an installation actually suffers from performance issues, a considerable amount of effort in system tuning is required.

The following text is a compilation from various forum posts and hints found elsewhere. Not all of the hints are fully tested yet.

9.1 Hardware considerations

- The machine which runs the ScientificCMSservlet container (and perhaps the database as well) should have at least 1gb of memory. To add more memory (2gb or even better 4 gb) is likely to increase the systems responsiveness. You have to perform additional configuration steps to be able to really use the added memory (see below).
- The database server (may be the same machine as the servlet container) should be equipped with a hardware raid controller and fast SCSI disks. ScientificCMSis disk intensive, so a fast disk system is essential.
- ScientificCMSdoes make heavy use of threads. So you can gain performance from a multiprocessor machine, even if the raw processor speed is slower than a single processor machine.
- Even in a small installation (but with a considerable amount of users / page hits) you can increase responsiveness by using separate boxes for the servlet container and the database server (a sort of primitive load balancing).

9.2 Tuning ScientificCMS

9.2.1 Increase the JDBC pool size

By default the JDBC pool size is set to 10. Increase it to 50 or 100. Use the following command:

```
ccm set waf.runtime.jdbc_pool_size=100
```

9.2.2 Increase the memory usable by the JAVA virtual machine

The amount of avaiable memory can be tuned by modifying the JAVA virtual machine configuration file. If you are using Red Hat Enterprise Linux (or a compatible) you should modify /etc/sysconfig/tomcatxxx file instead and adapt the tomcat runtime environment (root permission required). At the beginning of the file you find the following lines:

```
## Jave Runtime Environment parameters
# JAVA_OPTS=-Xms512m -Xmx512m
```

Uncomment the line beginning with JAVA_OPTS and adjust the values appropriately. If the machine has at least 2 gb RAM `JAVA_OPTS=-Xms512m -Xmx1024m` should be a good choice. Large values may have a negative effect on the garbage collection process. You should carefully check the effects of a modification!

If you use Suns Java JVM version 4.1 or later on a 32-bit system you should add the -server option (must be the first option in the list!)

```
## Jave Runtime Environment parameters
# JAVA_OPTS=-Xms512m -Xmx512m
JAVA_OPTS=-server
```

9.2.3 Explore and Adjust Caching mechanisms

If you use *Apache* (or any other Web Server) as your front end to the visitor (instead of using the servlet container's deploy facility) try caching by the web server. In case of Apache either use mod_proxy in reverse mode (preferred if you have just one application server) or mod_jk (Apache up to version 2.1 in case of a cluster of application servers).

9.2.4 Misc

There had been some discussion about adding a caching header control. This is still under investigation.

Others have found that PDF indexing is slowing things down. PDF indexing was eating up lots of memory and is worse-than-useless in context because search results don't indicate that the term looked for is in an attached PDF! If there are a lot of pdf attachments you may try as root from a command line

```
ccm set
com.arsdigita.cms.search.disableFileAssetExtraction=true
ccm stop
ccm start
```

You can inspect (but not modify) the settings on the page `/ccm/ds/config`.

9.3 Tuning Tomcat

- increase the maximum number of maxProcessor. Tomcat's default value is only 10 or 20 max. You can increase up to 300 or 500 (a number greater than 500 is reported to have a negative performance effect).
- increase timeout (accordingly)
- increase queuing, ie acceptCount

9.4 Tuning Postgres

9.4.1 Allocate more resources

- max_connections = 250
- shared_buffers = 2000
- max_fsm_pages = 100000

9.4.2 Reorganize the database nightly

You may prepare a script, named e.g. pg_maintenance in /var/lib/pgsql, to do the actual work. This script must be created and executed as user postgres (on Linux/Unix log in as user root and perform a su - postgres).

```
#!/bin/bash
#
# pg_maintenance
# script performs an backup of postgres databases and a
vacuumdb to
# reorganize the tables (tested with postgres version
7.4 on RHEL 4)
#
# adjust the following
BACKUP_DEST=/var/lib/pgsql/backups
# use this to keep a backup for a month
TIMEDUMP=`date +-%d`
# use this to keep every backup
#TIMEDUMP=`date +-%d-%m-%Y`
#
# usually you don't need to change anything beyond this
line
echo Starting backup ccm `date`  >> $
{BACKUP_DEST}/maintenance.log
pg_dump -Ft ccm -U postgres -b -f ${BACKUP_DEST}/ccm$
{TIMEDUMP}.tar
echo Finished backup ccm `date`  >> $
{BACKUP_DEST}/maintenance.log
echo `ls -l  ${BACKUP_DEST}/ccm${TIMEDUMP}.tar`  >> \
     ${BACKUP_DEST}/maintenance.log
echo Starting vacuumdb ccm `date`  >> $
{BACKUP_DEST}/maintenance.log
vacuumdb -d ccm -z -f              >> $
{BACKUP_DEST}/maintenance.log
echo Finished vacuumdb ccm `date`  >> $
{BACKUP_DEST}/maintenance.log
echo "--------------------------------" >>
${BACKUP_DEST}/maintenance.log
exit
```

Replace the database name (ccm in the example above) by the actual name of your ScientificCMSdatabase.

If you use a Linux or Unix box as your server you should use cron to perform the job. Prepare a template file for cron, named e.g. pg_crontab:

```
# pg_crontab
# postgres crontab file
# install as user postgres by
# crontab  pg_crontab
#
# don't mail any output
MAILTO=""
#
# run maintenance daily at 4:10 am
10 4 * * * /var/lib/pgsql/pg_maintenance
```

You can make it active by the command (being user postgres!)

```
crontab pg_crontab
```

You may list the current content of the crontab for user postgres by

```
crontab -l
```

You should check the log file regularly, esp. the amount of time the job will run. The script performs a full analyze every night. You may change it to a full analyse only once a week.

9.4.3 3.4.3 Try to change statistics for acs_objects

According to some findings a join query on the acs_object table sometimes causes performance problems. You could check if the following procedure results in an improvement:

```
ccm_perf=# ALTER TABLE acs_objects ALTER COLUMN
object_id SET STATISTICS 100;
ALTER TABLE
ccm_perf=# ANALYZE acs_objects;
ANALYZE
```

Caution

Postgres uses the general filesystem caching to buffer the files and for this reason there must always be free system memory on such systems. You have to adjust the memory allocation for the other components appropriately.

Helpful information about tuning Postgres can be found here:

- http://www.varlena.com/varlena/GeneralBits/Tidbits/perf.html
- http://archives.postgresql.org/pgsql-performance/2005-05/msg00210.php
- http://archives.postgresql.org/pgsql-performance/2005-05/msg00220.php

9.5 Tuning Oracle

Allocating memory may improve performance a lot.

```
db_cache_size=1073741824
pga_aggregate_target=805306368
shared_pool_size=469762048
sort_area_size=4194304
```

The actual figures depend on the total amount of RAM the server has! So you shouldn't use them as is, instead choosing an appropriate value for the sizes.

David Miller reports:

> ... the majority of the performance boost on the admin side was generated just by rebuilding the statistics for the user's tables. Logging in as system the sql to do this could be generated by running
>
> select 'analyze table aplaws_plus.' || table_name || ' estimate statistics;' from dba_tables where owner='[OwnerName]';
>
> The output from this query was then fed into a spool file and run to update the stats on every table.
>
> (put into oracle/../home/sql/buildstats.sql on apsun).

One of the best way to speed up to the DB is

```
exec
DBMS_STATS.gather_schema_stats('[YOUR_SCHEMA]',DBMS_STAT
S.AUTO_SAMPLE_SIZE);
```

HIGHLY recommended read: http://www.puschitz.com/TuningLinuxForOracle.shtml esp. note Huge Pages and filesystemio_options sections!

9.6 Tuning the Linux box

- increase maximum file handle for linux in `/proc/sys/fs/file-max`. e.g. 256xxx
- increase maximum file handle per process at `/etc/security/limits.conf`. The one with servlet soft and hard. eg. 32xxx
- increase maximum process per user via `ulimit -n VALUE` e.g. `ulimit -n 32xxx`
- increase `ulimit` to its maximum ... ie. `ulimit -u unlimited`
- other than that, you can also increase the range of open port if needed.

9.7 Tuning Squid

If Squid is used

- increase the cache_mem to something like 100 or 200 MB. You can increase more than that i guess.

9.8 Misc

Analyse the source of performance issues. The Developer Support with the query logging is really a great tool to see where you have problems in your application. (`http://<yourhost>/ccm/ds/`)

Still under investigation (quote from original postings):

- "we rewrote some of the core queries to avoid the join on acs_objects which can be very costly when you have a lot of objects in this table. Simply using a subquery instead of a join did the trick"
- "we implemented a CachedComponent object. Every component in our aplaws sites inherit from this component and we can cache every component for a certain amount of time in a CacheTable. This CacheTable can be managed through the DeveloperSupport interface if you have to remove a component from cache really quickly"

10. Extending an Installation

Probably you may wish to extend an ScientificCMS installation, either by adding new functionality, i.e. adding one or more applications, or by adding one or more ScientificCMS servers.

10.1 Adding New Functionality

For each installation bundle, there is an expansion kit (add-on modules) that contains all the modules that are not already in the installation bundle.

Download the appropriate expansion kit. If you have the Application Program Edition installed, the expansion kit is a RPM or DEB file and installed using the package manager. Otherwise it comes as a zip file and you may expand the zip file anywhere, probably in your home directory, but into its own subdirectory.

Then follow chapter 6.5 *Adding Packages to an Installed Bundle* on page 36.

10.2 Add Another Server

You may want to serve several different sites on your server. There are several options:

1. You may use the subsite module to make your installation respond to several addresses (e.g. www.mymusic.org and www.musicschool.org).

   Strictly spoken to don't create an additional ScientificCMS instance but make your current ScientificCMS installation available under various addresses, where you may provide a distinct navigation tree and site design for each and may provide distinct as well as shared content.

2. You may create several “virtual hosts” in your servlet container in install one or more ScientificCMS instances into each virtual host.

   Your ScientificCMS instance(s) inside a virtual host will be accessible under different domain names, as in alternative 1.

But all instances will be independent from each other and usually don't share the database (so can not deliver shared content).

10.2.1 Subsites

Refer to Administration Guide, chapter Subsite for configuration steps regarding ScientificCMS.

On the part of system administration and maintenance, a number of accompanying measures are required:

- For the sub-site, a DNS entry is required.
- The Tomcat host must be informed to handle an additional address using an Alias directive
- The Apache – if used as a front end – must be instructed to serve an additional address using an Alias directive.

Tomcat Configuration

You have to modify the file `server.xml` either in the `/etc/scientificcms` or in the `/etc/tomcat[x]` directory, depending whether the Application Program Edition or the WAR File Edition is installed.

As an example:

```
<Host name="www.domain1.com" debug="0"
      appBase="webapps" unpackWARs="true">
     <Alias>www.anotherdomain.com</Alias>
     . . .
</Host>
```

Apache Configuration

You have to add a Alias directive to the (virtual) host configuration:

```
<VirtualHost *>
    ServerName www.domain1.com
    ServerAlias www.anotherdomain.com
    DocumentRoot /var/tomcat4/webapps/domain1
    JkMount /* ajp13
</VirtualHost>
```

Apache will forward requests to both domains to the same Tomcat instance which in turn serves both addresses in the same (Tomcat) host and therefore using the same ScientificCMS instance.

At http://help.hardhathosting.com/question.php/150 you will find additional explanations. A hightly recommended tutorial can b e found ad http://www.csse.uwa.edu.au/~ryan/tech/tomcat.html

10.2.2 Additional virtual hosts

Creating Separate Servlet Container Instances

This option involves standard administration tasks. You first create one or more servlet container instances on your server and than perform a standard APLAWS+ installation for each one. As an example we describe the process for Tomcat on Red Hat Enterprise Linux.

Separate Tomcat Instances in Red Hat Enterprise Linux

Red Hat Enterprise Linux (RHEL) uses jpackage.org deployment system which is prepared to provide additional Tomcat instances where each run in a separate Java virual machine (JVM).

Create a new database user and database for the new instance

In most cases you will not only use a separate database but also a separate database user account.

For PostgreSQL you have to become the user postgres and execute the commands to create a new user and a database:

```
root@localhost# su - postgres
postgres@localhost$ createuser  -S -R -D -P ccmnew
Enter a password for the new role: ccmnew_xyz
postgres@localhost$ createdb -E UNICODE -O ccmnew ccmnew
postgres@localhost$ createlang plpgsql ccmnew
postgres@localhost$ exit
root@localhost#
```

Create a new ccm instance

```
root@localhost# ccm  mkservice /var/lib/ccmnew   ccmnew
You can now load your CCM instance "ccmnew" using:
# CCM_HOME=/var/lib/ccmnew ccm load ...
# CCM_HOME=/var/lib/ccmnew ccm hostinit ...
Start using:
# service ccmnew start
```

The script creates the necessary directory structure in /var/lib as well as a new start script in /etc/init.d so the new instance can automatically started at system boot.

You can now use the usual ccm commands by prefixing them with the corresponding environment variable.

Load the APLAWS+ database schemes and data

This is the usual step to prepare the database for usage by APLAWS+ (see Installation Guide chap. 4.2.2 for details). You must choose a distinct port number for the application container to run. The standard container uses port 8080 (and 8081 for the shutdown procedure), so you may use 8082 (and 8083) for your new instance.

```
root@localhost# CCM_HOME=/var/lib/ccmnew ccm load-bundle
\
            --interactive --name aplaws-plus-standard
```

Fill in the form for database access, server address, etc. as described in the installation manual and adjust the port numer properly.

Create the servlet webapp and configuration files

This step will copy all necessary files into a webapps directory (/var/lib/ccmnew in this example), which will be used by the new APLAWS+. The port number you specify has to be the same as provided to the ccm load command previously. Additionally you have to provide a new unique ajp-port.

```
root@localhost# CCM_HOME=/var/lib/ccmnew ccm hostinit-
bundle --clean \
            --name aplaws-plus-standard
--container=tomcat \
            http-port=8082 shutdown-port=8083 ajp-
port=8010
```

Start the server instance and check

```
root@localhost# service ccmnew  start
```

or

```
root@localhost# CCM_HOME=/var/lib/ccmnew ccm start
```

The log files for this instance are located at /var/lib/ccmnew/logs .

Point a web browser to localhost on port 8082.

```
root@localhost# links http://localhost:8082/
```

If you use Apache to deliver your Web Information you have to integrate the new instance into Apache (see chapter 2, page 7 for details).

11. Troubleshooting

java.lang.OutOfMemoryError: PermGen space

Description

The servlet container stops working and you find in the log file a message: „java.lang.OutOfMemoryError: PermGen space”

Solution

Enlarge the PermGen space of the Java VM. Add to JAVA_OPTS “-XX:MaxPermSize=256m”. Default is 64m. Eventually 128m may be sufficient of even 512m is needed.

Comment

The problem is likely to occur if you are running multiple large web applications in one servlet container. While deploying the classloader loads all the libraries into the PermGen space (permanent heap generation) of the JVM. Also, if you redeploy quite often (as on a development system) the default memory space may be insufficient.

In general it is a symptom of an incomplete garbage collection sweep (generational garbage collection), where resources are not properly released upon unload / restart.

See: Tomcat Wiki OutOfMemory

Appendix

Appendix 1: Installing the Java Development Kit on Linux

On Linux you have two options for installation of a JDK

3. You can use the Java vendor-specific installation, which usually installs all the component as a set of subdirectories under one main directory (usr/java/jdk1.6.v_rr in case of Sun/Oracle). This is straightforward, but is not compliant with the standard layout of the Linux File System Hierarchy and there is no standard path for third party Java libraries, which sometimes makes the construction of the classpath difficult.
4. You can install according the layout developed by the jpackage.org initiative. The various types of files (binary, lib, documentation, etc) are nicely integrated into the Linux File System Hierarchy and there are defined places for library files. It provides a sound installation framework for any Java software.

   The Open Source Java JRE / JDK OpenJDK uses this installation schema.

You can even mix both types of installation (i.e. using a vendor's installation for the Java SDK and a jpackage.org installation for a Tomcat servlet container). Jpackage provides compatibility links.

It is possible to use both types of installation. On a Red Hat system a jpackage-based installation is preferred (Red Hat is strongly engaged in the jpackage project and has integrated this way of installation into the distribution). All the java packages, which are delivered with a Red Hat distribution, follow the jpackage.org file organization principles.

Further information is provided by the jpackage.org home page (http://www.jpackage.org)

Appendix 2:
Setting up a PostgreSQL Database

It is strongly recommended to **use PostgreSQL 8.x**. Older version of PostgreSQL will not work reliably with version 2.x.

On a Red Hat compatible distribution the packages listed beyond must be installed:

```
postgresql-8.m.x-x.yyy
postgresql-server-8.m.x-x.yyy
postgresql-pl-8.m.x-x.yyy
postgresql-libs-8.m.x-x.yyy
postgresql-docs-8.m.x-x.yyy
postgresql-jdbc-8.m.xxx-xjpp.y
```

For other distributions or operating systems check for the equivalent packages.

Your PostgreSQL configuration must allow TCP/IP connections. By default some distributions allow socket connections only (cf. type 'host' in the listing below is commented out). On Red Hat Linux and compatibles it is enabled in version 8.x.

To avoid database connection problems during the installation, you may enable a relaxed authentication for installation as well.

Edit the file /var/lib/pgsql/data/pg_hba.conf and check or modify the last lines as follows:

```
# PostgreSQL Client Authentication Configuration File
# ===================================================
#
# Refer to the PostgreSQL Administrator's Guide, chapter
"Client
# Authentication" for a complete description.  A short
synopsis
# follows.
#
.........
.........
# CAUTION: Configuring the system for local "trust"
authentication
# allows any local user to connect as any PostgreSQL
user, including
# the database superuser. If you do not trust all your
local users,
# use another authentication method.

# TYPE  DATABASE    USER        CIDR-ADDRESS
METHOD
# "local" is for Unix domain socket connections only
local   all         all
trust
# IPv4 local connections:
host    all         all         127.0.0.1/32
trust
# IPv6 local connections:
host    all         all         ::1/128
trust
```

Database performance is critical for the overall performance of the system because it is very database centric and you should consult the postgreSQL documentation for performance tuning. As a first step you should edit the file ~/pgsql/data/postgresql.conf and modify the parameters *max_fsm_pages* and *max_fsm_relations* from their default values, which are just enough. Reasonable values are

```
max_fsm_pages = 30000
max_fsm_relations = 1500
```

You have to restart postgreSQL after any modification of the file (on Red Hat: 'service postgresql restart').

You may use the authentication method 'trust' during the installation to minimize the likelihood of errors. When the system is up and running, you should switch to a more secure method which is appropriate to your systems usage.

Check if PostgreSQL will start automatically at system boot. On Red hat you can use:

```
root@localhost# chkconfig --list postgresql
```

For other distributions use the appropriate command or check /etc/rc.d. If not, use the appropriate command to activate the server. On Red Hat or compatibles use

```
root@localhost# chkconfig --level 35 postgresql on
```

If you refer to install the database on a separate server, proceed accordingly. In addition you have to ensure

- the tcp/ip port for the postgresql clients (5432 by default, configured at compile time) is not blocked by a running firewall
- enable not-local access in postgresql.conf file

Appendix 3:
Installing Tomcat 6 on Linux

As with Java (cf, Chapter XX) there are 2 installation options: You may install

5. the Apache distribution, provided by apache.org, which may be installed into any directory
6. the jpackage.org distribution, which follows the Linux file system hierarchy

If there is a newer jpackage.org version of Tomcat already installed on your system, you must use the vendor distribution. Different versions can not be installed in parallel.

If you have to use non-ASCII characters in your content pages, a jpackage.org based installation sometimes suffers from erroneous reactions in the editor pane. It replaces some libraries which used to be part of the internal lib classpath by systemwide libraries.

As an alternative you may install a Tomcat as provided by the APLAWS project which combines the advantages of a jpackage.org installation with the self-contained principle of vendor's installation.

3.1 Installation of the vendor distribution

- Log in to your system as root and switch to the intended installation directory, either /opt or /usr/local
- Download the installation file from http://tomcat.apache.org/download-60.cgi
- Unpack the distribution file

```
root@localhost # tar xzvf apache-tomcat-6.0.rr.tar.gz
```

Replace rr by the release number. As of this writing it is 32.

You will find a subdirectory apache-tomcat-6.0.rr in your installation directory when tar has finished. For convenience you may add a symbolic link:

```
root@localhost # ln -s apache-tomcat-6.0.rr tomcat
```

- Place a file named “tomcat.sh” in the directory /etc/profile.d with the following instructions:

```
#  /etc/profile.d/tomcat.sh
#  Setting environment variable for Tomcat
TOMCAT_HOME=$CATALINA_HOME
export TOMCAT_HOME
```

  You have to ensure that the TOMCAT_HOME environment variable is set.

  Log out of your machine completely. If you are on the console, exit back to the graphical login screen. If you are remote, then

  

  If you do not logout completely, your shell session will not have any of the required global environment variables set. This will cause later commands to fail.

  terminate your SSH session.

3.2 Installation of a jpackage.org distribution

3.2.1 Pre-requisites

In order to use the jpackage.org RPMs you have to install

- jpackage-utils
- update-alternatives

On a recent Red Hat distribution both packages should already be installed. Otherwise download and install the RPMs from jpackage.org.

3.2.2 Getting and Installing the Software

You will find Tomcat 4 at

http://mirrors.dotsrc.org/jpackage/5.0-updates/generic/free/repoview/tomcat6.html

You will need the following packages:

- tomcat6-6.0.rr-xjpp5.noarch.rpm

Those RPMs depend on a lot of other RPMs. You have three options to proceed:

- Download the above mentioned Tomcat 6 package into a separate directory. Open a terminal window, switch to the download directory and use

```
root@localhost# rpm –i –-test tomcat6*
```

  to determine the needed packages and download them from the jpackage site. Be careful to pick up the correct xml/xslt packages (xerces-j2/xalan-j2). Then use

```
root@localhost# rpm –i *.rpm
```

  as root to install the packages.

- If you have yum installed you can use it to download and install Tomcat 6 from jpackage.org. Yum will automatically resolve all dependencies. You will find instructions for configuring yum on the jpackage.org site. Yum may download the crimson package for xml/xslt handling. It does not work for ScientificCMS You have to download and install xerces-j2 and xalan-j2 manually instead before installing ScientificCMS.

11.1 Additional Libraries

Just in case you use Xerxes of SAY XML interpreter you *may* have to install additional libraries in Tomcat's lib directory. This requirement is indicated by error messages in the log file missing XML specific classes.

11.2 Multiple Tomcat Instances

It is possible to create multiple Tomcat instances with the use of the CATALINA_BASE environment variable. Each instance uses a common binary distribution but uses its own conf, webapps, temp, logs and work directories. Each instance also has its own JVM and, thereby, its own memory pool. If you have defined the maximum memory to be 512MB via JAVA_OPTS, each instance will attempt to allocate a maximum of 512MB.

Additional required edits are to change the SHUTDOWN port from 8005 to 8105.

Now we must change the AJP connector from 8009 to 8109.

If you are using SSL, you also should change the redirectPort to the appropriate SSL port that Apache is listening on, normally 443.

11.2.1 Tomcat original standard distribution

Let's proceed now to set up these directories. As I mentioned before, Tomcat is installed in /opt/tomcat. To keep things somewhat organized, I created the following folders in /opt: /opt/tomcat_instance1, /opt/tomcat_instance2 and /opt/tomcat_instance3. It probably is more appropriate, however, to name these folders based on their purposes or applications. Remember that each of the three folders will contain conf, webapps, temp and work directories

11.2.2 Step-by Step Procedure

1. Choose a (self-explaining) name for your additional instance.

   The default name for the first (primary) instance is tomcat6. So you may choose names like tomcat6-customerX or tomcat6-myapps. We use *tomcat6-hosted* as an example in this step-by-step guide.

2. In /etc/init.d (may be different with ohe Linux distributions) create a link named like your choosen name to the original init script tomcat6. Create a **link**, not a copy!

   ```
   root@localhost# cd /etc/init.d
   root@localhost# ln -s tomcat6 tomcat6-hosted
   ```

3. Create a subdirectory for configuration files

   ```
   root@localhost# cd /etc/
   root@localhost# mkdir tomcat6-hosted
   root@localhost# cp -a tomcat6/*  tomcat6-host/
   root@localhost# cd tomcat6-hosted
   root@localhost# rm tomcat6.conf
   ```

4. Create a copy of the system configuration file and adopt it

   ```
   root@localhost# cd /etc/sysconfic
   root@localhost# cp tomcat6  tomcat6-hosted
   root@localhost#
   ```

5. xxxxx

6. Create a Directory for the instance specific log files

```
root@localhost# cd /var/log
root@localhost# mkdir tomcat6-hosted
root@localhost# chown tomcat6.tomcat6  tomcat6-hosted
root@localhost#
root@localhost#
```

7. Create base directory for the instance specific temporary files

```
root@localhost# cd /var/cache
root@localhost# mkdir tomcat6-hosted
root@localhost# mkdir tomcat6-host/work
root@localhost# mkdir tomcat6-host/temp
root@localhost# chown -R tomcat6.tomcat6  tomcat6-hosted
root@localhost#
root@localhost#
```

8. Create a webapps directory for the additional instance if appropriate (i.e. if you don't use a (virtual) host specific directory

```
root@localhost# cd /var/lib
root@localhost# mkdir tomcat6-hosted
root@localhost# mkdir tomcat6-hosted/webapps
root@localhost# chown -R tomcat6.tomcat6  tomcat6-hosted
root@localhost#
```

9. Create the base directory for the additional instance

```
root@localhost# cd /var/lib
root@localhost# mkdir tomcat6-hosted
root@localhost# cd tomcat6-host/work
root@localhost# ln -s conf /etc/tomcat6-hosted
root@localhost# ln -s log /var/lib/tomcat6-hosted
root@localhost# ln -s temp /var/cache/tomcat6-
hosted/temp
root@localhost# ln -s work /var/cache/tomcat6-
hosted/work
root@localhost#
root@localhost#
```

10. xxxxx
11. xxxxxx

11.2.3 Configuring Apache mod_jk

mod_jk uses a file named workers.properties. I recommend placing this file with the rest of your Apache configuration files. workers.properties is used to define where Apache looks for the Tomcat instances. Here, I cover only the items we are going to use for the three instances we have set up. Below is the workers.properties file we are going to use, followed by an explanation of the options:

If you use an apache integration mod_jk you have to add an additional worker entry for the newly created Tomcat instance:

```
root@localhost# cd /etc/httpd/conf
root@localhost# vi worker.properties
root@localhost#
```

You will find something like this:

```
worker.list=worker1,worker2,worker3
# Set properties for worker1
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
# Set properties for worker2
worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8109
# Set properties for worker3
worker.worker3.type=ajp13
worker.worker3.host=localhost
worker.worker3.port=8209
```

Modify the file according to the following guidelines:

worker.list is a comma-separated list of worker names. You could have Tomcat workers defined later in the file that will not be used. Any worker defined is not used unless the worker is listed in the worker.list value. Workers are defined in format of `worker.NAMEOFWORKER.type`, with the value being the type of connector. All of our workers are of type ajp13. In the above example, we have defined three workers: worker1, worker2 and worker3.

- There is a line beginning with
- Add a triple set of lines

11.2.4 Configuring Apache to Forward

We use domain1, domain2 and domain3 as our virtual hosts. To pull this off, we also have to edit /etc/hosts to ensure that domain1, domain2 and domain3 are resolved properly. To do this, we make three VirtualDirectory declarations, each corresponding to a worker defined in workers.properties. Below is the VirtualHosts section, followed by an explanation of the options.

```
<VirtualHost *:80>
    ServerName domain1
    JkMount  /servlets-examples/* worker1
</VirtualHost>
<VirtualHost *:80>
    ServerName domain2
    JkMount  /servlets-examples/* worker2
</VirtualHost>
<VirtualHost *:80>
    ServerName domain3
    JkMount  /servlets-examples/* worker3
</VirtualHost>
```

see also:

http://bderzhavets.blogspot.com/2006/07/advanced-configuration-multiple-tomcat.html

http://permalink.gmane.org/gmane.linux.jpackage.general/14109