OSAAS Documentation

Release 1.0.0

Ingo Weinzierl

January 26, 2011

CONTENTS

Contents:

CHAPTER

ONE

ABOUT *OSAAS - OWS STATISTICS AND ACCOUNTING SYSTEM*

OSAAS contains of 2 components, the server and clients. One client for deegree OWS-Proxy is part of this package. Arbitrary others could be implemented.

OWS stands for OGC Web Services where OGC is the Open Geospatial Consortium.

OSAAS is Free Software, most of it licensed under GNU GPL (the server). Other parts depend on the respective clients (e.g. GNU LGPL for deegree).

Please refer to the README.txt in the respective sub-directories for more information.

For further questions or discussion please get in contact via the delelopers mailing list:

http://lists.wald.intevation.org/mailman/listinfo/osaas-devel

CHAPTER

TWO

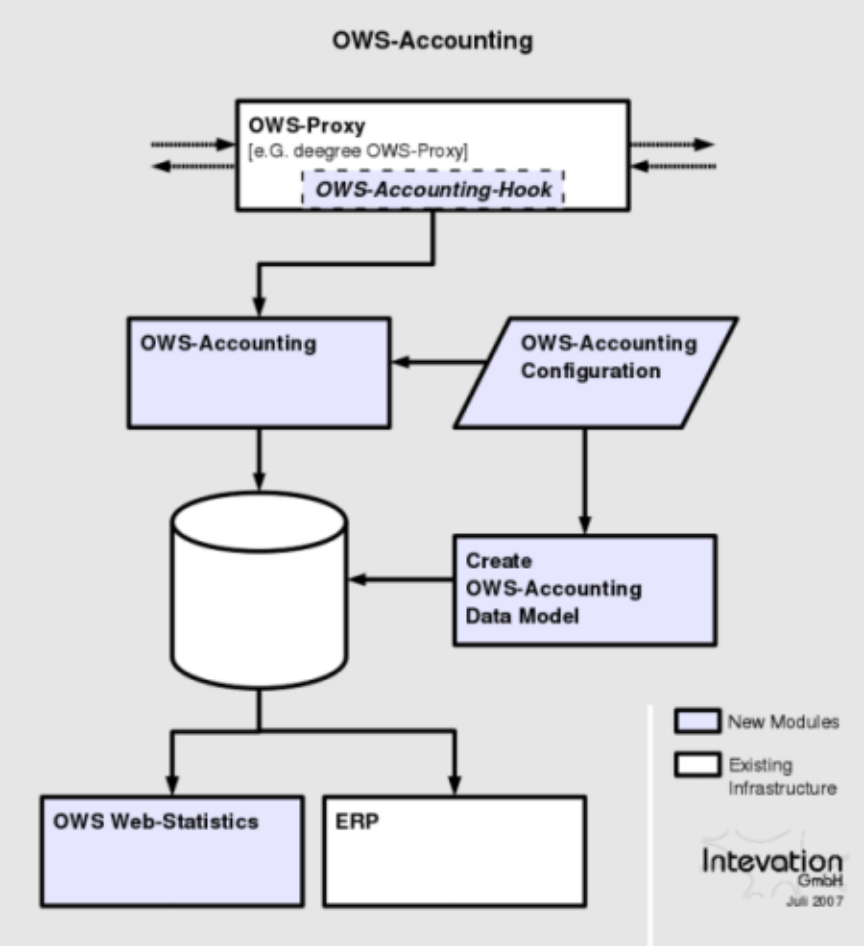
GOALS AND CONCEPTS

The main goal of *OSAAS* is to track access to your OGC web services for the purpose of statistics. In addition to that, its integrated user authentication enables you to take account for accessing OGC services.

For this reason, an *OSAAS* client (e.g. deegree OWS-Proxy with *OSAAS* hook) communicates with an *OSAAS* server. The client works as proxy in front of the actual OWS. It recieves the OGC requests and forwards them to the actual OWS, as well as it sends information about each of those requests to *OSAAS* server. Currently, the only way of communication between client and server takes place via HTTP-POST using HTTP-BasicAuth to authenticate *OSAAS* clients.

Note: The goal of *OSAAS* user authentication is to grant an *OSAAS* client access to *OSAAS* server. It is absolutely independent from the user authentication of OWS-Proxy.

Finally, *OSAAS* server logs the OGC requests to a database, that makes these information available for other tools - *OWS Web-Statistics* is one of this tools.



Some tools of *OSAAS* server have been created with the aim to make the installation and configuration as easy as possible. For example, there is a tool that creates new users, another one creates the whole data model based on the configuration file. So it is not necessary for the user to know about the database schema in detail. For more information see *OSAAS Installation*.

CHAPTER

THREE

INSTALLING OSAAS SERVER

The OSAAS server is the core component of OSAAS. It accepts HTTP requests from OSAAS clients containing information about WMS requests, processes the information and stores it into a database. This chapter will lead you through OSAAS installation process.

3.1 Requirements

Before you are able to install *OSAAS*, you need to install some software:

- Python >= 2.3 (see http://python.org/)

Depending on which database you use, you need one of the following database API modules:

- cx_Oracle >= 4.3.1 (see http://www.python.net/crew/atuining/cx_Oracle/)
- psycopg2 >= 2.0.5.1 (see http://www.initd.org/pub/software/psycopg/)

After downloading and installing the required software, you should download *OSAAS* from wald.intevation.org or using *Subversion* (see http://svnbook.red-bean.com/) with the following command:

```
svn checkout https://svn.wald.intevation.org/svn/osaas/trunk osaas-trunk
```

Note: The recommended way of getting *OSAAS* is to download the tarball of an official release instead of checking out the development branch via SVN.

3.2 PostgreSQL Configuration

The next step is to set up your PostgreSQL database. If you have installed PostgreSQL from Debian packages, the path definitions of the next steps should go with your environment. Otherwise, the pathes may differ.

At first, you should create a user *osaas* and the database *osaas_logging* with the commands below:

```
su postgres -c "createuser -SDRleP osaas"
su postgres -c "createdb -O osaas osaas_logging"
```

In order that the created user is allowed to access the database, it is necessary to modify `/var/lib/pgsql/data/pg_hba.conf` and `/var/lib/pgsql/data/postgresql.conf`.

In `/var/lib/pgsql/data/postgresql.conf` you will find the config options *listen_adress* and *port*. Modify these as follows:

```
listen_addresses='localhost'
port = 5432
```

and append the following line to `/var/lib/pgsql/data/pg_hba.conf`:

```
host osaas_logging osaas  127.0.0.1/32 md5
```

Now, the PostgreSQL database is ready. Restart it with the command:

```
/etc/init.d/postgresql restart
```

3.3 OSAAS Server Installation

OSAAS server is written in python and needs no explicit installation. Downloading the tarball from wald.intevation.org or via SVN is all you need to do. But there are some things you need to initialize before you are able to run *OSAAS*.

First of all: *OSAAS* server requires a user authentication. Use the `adduser.py` script to create new users. The following command will create a new user *osaas* and ask you for a password for this user. Afterwards, the user and its hashed password are stored in the user database file given with the -f option. In this case, users and passwords are stored in *my-osaas-users*:

```
./server/adduser.py -f my-osaas-users osaas
```

OSAAS itself reads a configuration file that contains the database connection parameters, server settings and information about *OSAAS* datamodel. `server/demo-config.xml` is a demo configuration. See Configuration for more details on possible configuration options.

After creating a new user and a configuration file, you are ready to initialize the database using `initosaasdb.py`. `initosaasdb.py` reads your configuration and creates all necessary database tables - independent of which database (Oracle or PostgreSQL) you use. Initialize the database with:

```
./server/initosaasdb.py --config-file=my-osaas-config.xml
```

If everything was fine, you can start *OSAAS* server (see *Start/Stop OSAAS Server*).

CHAPTER

FOUR

CONFIGURATION

As already mentioned above, you can find a demo configuration in `server/demo-config.xml`. To understand the config options, please read the comments inline this configuration:

```
<?xml version="1.0"?>
<OSAASConfig>
  <!-- The port the server will listen on -->
  <Port>8989</Port>

  <!-- The files the server will log to.  The access log holds the usual
       HTTP server access log information.  The error log is used to log
       errors and debug information.  If no file is specified, the
       logging output is written to stderr.

       Supported log levels in order of increasing Verbosity:
           CRITICAL, ERROR, WARNING, INFO, DEBUG
       Default log level is INFO.
    -->
  <AccessLog>/var/log/osaas/osaas-access.log</AccessLog>
  <ErrorLog>/var/log/osaas/osaas-error.log</ErrorLog>
  <LogLevel>INFO</LogLevel>

  <!-- The users database file. -->
  <UserDB>/etc/osaas/osaas-users</UserDB>

  <!-- File in which to write the PID.  Useful for /etc/init.d scripts -->
  <PidFile>/var/run/osaas.pid</PidFile>

  <!-- The database connection parameters -->
  <DBConnection>
    <!-- Type of the database.  Possible values: oracle, postgresql -->
    <Type>postgresql</Type>

    <!-- Connection parameters as a single string.  Use this for postgresql. -->
    <Parameters>user=mydbuser password=mydbpassword host=localhost port=myport dbname=mydb</Parameter

    <!-- Connection parameters for Oracle.  Make sure Parameters is
     empty or not present at all when using oracle.  The Parameters
     entry takes precedence over the following parameters. -->
    <User>mydbuser</User>
    <Password>mydbpassword</Password>
    <DSN>myDSN</DSN>
  </DBConnection>

  <!-- The database rules.  The rules describe one table and a mapping
```

```
       from the values sent to the OSAAS server to the columns in the
       table.  Some of the values are generated by the OSAAS client (the
       WMS server or OWS proxy) and some are taken from the original
       request sent by the WMS client.  All information taken from the
       original WMS request has names in UPPERCASE.  The values
       generated by the OSAAS client have lowercase names.
    -->
  <DBRules>

    <!-- Definition of the MyOWSAccounting table.  The server expects
     this table to have all the columns defined here and an
     additional column called ID with a unique ID generated by a
     sequence called MyOWSAccounting_seq (the name is derived from
     the table name by append "_seq"). -->
    <insert table="MyOWSAccounting">

      <!-- Defines the column wmsidintern of type character varying(100)
       which will hold the wmsidintern value -->
      <value param="wmsidintern" column="wmsidintern"
         type="character varying(100)"/>

      <!-- external wmsid, defined like wmsidintern -->
      <value param="wmsidextern" column="wmsidextern"
         type="character varying(100)"/>

      <!-- the name of the user is stored in the username column -->
      <value param="user" column="username" type="character varying(20)"/>

      <!-- The start and end timestamps -->
      <value param="starttime" column="starttime"
         type="timestamp with time zone"/>
      <value param="endtime" column="endtime"
         type="timestamp with time zone"/>

      <!-- The values taken from the original WMS request-->
      <value param="SERVICE" column="service" type="character varying(10)"/>
      <value param="REQUEST" column="req" type="character varying(10)"/>
      <value param="BBOX" column="bbox" type="character varying(100)"/>
      <value param="WIDTH" column="width" type="int"/>
      <value param="HEIGHT" column="height" type="int"/>
    </insert>
  </DBRules>
</OSAASConfig>
```

CHAPTER

FIVE

BUILDING OSAAS CLIENT

OSAAS Java Client is a small library for java programs that want to send data to the OSAAS server. The build process uses ant. So, you need to make clear that you have ant installed (http://ant.apache.org/).

Now, if you have ant installed on your machine, you can change to the `client/java/` directory and enter the following command:

```
ant
```

The default ant target builds a `osaas-client.jar` file in the `dist/` subdirectory. Use this file in your client.

5.1 Integration in *deegree OWSProxy*

First of all, you need to download the deegree2 sources in a specific version, apply the patch `client/java/owsproxy/owsproxy.diff` and make *OSAAS* client available to deegree. After this, you are ready to build *deegree OWSProxy*. Follow the instructions below:

1. Download the sources using deegree:

   ```
   svn co -r 15164 https://svn.wald.intevation.org/svn/deegree/base/trunk deegree-osaas
   ```

2. Apply the patch `client/java/owsproxy/owsproxy.diff`:

   ```
   cd deegree-osaas
   patch -p 0 < owsproxy.diff
   ```

3. Make the osaas client code available to deegree (`$DIST_HOME_OSAAS_CLIENT` is the directory where `osaas-client.jar` is located):

   ```
   cd deegree-osaas/lib
   ln -s $DIST_HOME_OSAAS_CLIENT/ osaas
   ```

4. Build deegree:

   ```
   ant build-lib
   ```

Now, you have built *deegree OWSProxy* with *OSAAS* integration. Finally, you need to configure logging. *OSAAS* logging configuration is read from the same WEB-INF/web.xml file as the *OWSProxy* configuration. Add the following lines to this file and replace the url, username, password and faillog:

```
<!-- The URL to which the OSAAS logging information is to be posted
     Note: The url has to end with a '/'.
-->
<init-param>
```

```
   <param-name>OSAAS_URL</param-name>
   <param-value>http://localhost:8989/owsaccounting/</param-value>
</init-param>

<!-- The user name and password to use for authentication with the
     OSASS server -->
<init-param>
   <param-name>OSAAS_USERNAME</param-name>
   <param-value>aUser</param-value>
</init-param>
<init-param>
   <param-name>OSAAS_PASSWORD</param-name>
   <param-value>secret</param-value>
</init-param>

<!-- The file into which failed logging attempts are written -->
<init-param>
   <param-name>OSAAS_FAILLOG</param-name>
   <param-value>/usr/share/tomcat/logs/osaas-faillog</param-value>
</init-param>
```

CHAPTER

SIX

START/STOP OSAAS SERVER

Starting and stopping of *OSAAS* server depends on your installation. This chapter describes the two most frequent ways.

6.1 Init script

If you want to start *OSAAS* server using the init script, you need to pay attention of two things:

- the script requires your configuration at `/etc/osaas/osaas-config.xml`
- the script needs to have executable permissions
- the script requires an osaas-user to run under

If both points are clear, you are able to start *OSAAS* server with the following command:

```
/etc/init.d/osaas start
```

To stop *OSAAS* server, just type:

```
/etc/init.d/osaas stop
```

6.2 Start/Stop *OSAAS* from sources

To start *OSAAS* server from sources, you will have to use the `startosaas.py` script. Change to the directory where the script is located and type:

```
./startosaas.py --config-file=your-config.xml
```

This line starts *OSAAS* server with a configuration located in `your-config.xml`. Note, that all of the settings from config file, except of the DBRules, can be specified on the command line as well. You can find a list of all available command line options below:

–config-file=CONFIG_FILE Use this option to specify the osaas configuration that should be used to run *OSAAS*.

–port=PORT Use this option to specify the port the *OSAAS* server will listen on.

–access-log=ACCESS_LOG This option specifies the destination of the access log. *ACCESS_LOG* is the path to a file. Note, that the user that runs *OSAAS* needs to have write permissions for this file.

–error-log=ERROR_LOG This option specifies the destination of the error log. *ERROR_LOG* is the path to a file. Note, that the user that runs *OSAAS* needs to have write permissions for this file.

–pid-file=PID_FILE This option specifies the file where to write the PID.

–log-level=LOG_LEVEL This option specifies the detail level of the logging. Supported log levels are *CRITICAL*, *ERROR*, *WARNING*, *INFO* and *DEBUG*.

–userdb-file=USERDB_FILE This option specifies the user database file that is used for the user authentication of *OSAAS*. *USERDB_FILE* is a path to a valid *OSAAS* user database file.

–db-type=DB_TYPE This option specifies the database type. Currently, the supported types are *oracle* and *postgresql*.

–db-parameter=DB_PARAMETERS This option specifies the parameters used to connect to a PostgreSQL database. *DB_PARAMETERS* is a single string.

–db-user=DB_USER This option specifies the user that is used to connect to a Oracle database.

–db-password=DB_PASSWORD This option specifies the password that is used to connect to a Oracle database.

–db-dsn=DB_DSN This option specifies the dsn that is used to connect to a Oracle database.

To stop *OSAAS* server, just kill the process (Ctrl+C). *OSAAS* will stop listening in the socket, wait until any pending requests have been processed and then terminates.

CHAPTER

SEVEN

CONTRIB

Currently, the contrib directory contains two scripts - a python script named *billing.py* and an init script named *osaas*.

7.1 Init script

osaas is a SuSE-specific init-script which adds the possibility to start and stop OSAAS-server as a service. You can manually copy it to `/etc/init.d/` and set the propper links to the runlevels you want to start the OSAAS-server. By default this script requires an osaas-user to run under. Make sure that the script has the executable-permissions set. The startup-script requires a configuration-file `/etc/osaas/osaas-config.xml` to work properly. When using OSAAS-server from within a RPM-package this is automatically setup correct.

7.2 billing.py

billing.py searches for GetFeature requests in the database where OSAAS is logging to. Each GetFeature request that could be found is sent to the WFS service again - with a little difference: the script doesn't query for the features itself, but for the number of features that would have been retrieved by this GetFeature request (the *resultType=hits* parameter is set in this GetFeature request). Finally, the retrieved number is written into the database.

The next sections describe the configuration and usage of *billing.py*.

7.2.1 PostgreSQL Configuration

billing.py needs access to the database where OSAAS is logging to. So, make sure that PostgreSQL is listening to localhost on port 5432. Therefore, you should find the following lines in `/var/lib/pgsql/data/postgresql.conf`:

```
listen_adress='localhost'
port = 5432
```

If OSAAS is logging to a database which is not running on localhost or port 5432, you need to adjust these settings.

Furthermore, you should find a line to allow the user *osaas* to access to *osaas_logging* database running on localhost in `/var/lib/pgsql/data/pg_hba.conf`:

```
host osaas_logging osaas  127.0.0.1/32 md5
```

Now, you need to create the user and the database:

```
su postgres -c "createuser -SDRleP osaas"
su postgres -c "createdb -O osaas osaas_logging"
```

Finally, you need to execute two SQL scripts that create all necessary database tables. The first one creates the target database table for WFS requests. The second one adds some additional columns to this table required by the billing script:

```
su postgres -c "psql -U osaas -h localhost -d osaas_logging -f contrib/billing/wfs_requests.sql"
su postgres -c "psql -U osaas -h localhost -d osaas_logging -f contrib/billing/billing.sql"
```

7.2.2 Configuration File

This section describes the options, the user has to configure the *billing.py* script. In general, the config is devided into two parts - database and logging. You can find an example configuration *billing.cfg* in `contrib/billing/`.

[database]

The settings in this section define the database connection. It is absolutely necessary to have such a section in your configuration - otherwise *billing.py* is not possible to create a connection to the database *OSAAS* is logging to.

The options of this section are:

database (required): This option defines the name of the database where *OSAAS* is logging to. It is absolutely necessary to define this option - otherwise the script wouldn't be able to connect to a database.

host (optional): This option defines the host where the database is running on. If this option is missing, its default value is *localhost*.

user (required): This option defines the user that is used to connect to *database*. It is absolutely necessary to define a user - otherwise the script wouldn't be able to connect to database.

password (optional): This option defines the password for *user* that is used to create the database connection. If there is no password in the configuration, the database connection is created without password.

[logging]

The settings in this section define the logging of *billing.py*. All options in this section are optional.

The options in this section are:

logfile (optional): A path to a file that is used for logging. It is recommended to define this option to get to know about important information of the execution of the script.

loglevel (optional): This option defines the level of details. Currently, you are able to choose between three levels - *1*, *2* and *3*. The default loglevel is *2*. *1*: The script just logs failures that occured while processing. *2*: The script logs failures and some additional information. *3*: The script is very verbose - logs failures, additional information and a lot of further information about the process itself. This loglevel is recommended for developers.

CHAPTER

EIGHT

UTILIZATION

This section will give you some example sql statements that might help you to list all requests of a specific user in a specific interval.

You can execute the queries below, e.g. when you are logged in the psql terminal (see the documentation for more information about psql):

- List all requests of user *my-user* in the last 7 days:

```
SELECT * FROM myowsaccounting WHERE starttime > (current_timestamp - interval '7 days') AND us
```

- List all requests of user *my-user* in the last month:

```
SELECT * FROM myowsaccounting WHERE starttime > (current_timestamp - interval '1 month') AND u
```

CHAPTER

NINE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*