# Grouper book - Installation of core components

## Grouper is now containerized and this document is in reference to the legacy installation process.

- Install the Grouper container with maturity level 0
- Install the Grouper container with maturity level 1

## Setting up core Grouper components - database and Daemon

We will be installing Grouper using the individual components, rather than using the bundled quick start. This is not because the quick start lacks any power or functionality (in fact it contains the entire API and UI), but because installing them individually gives you more flexibility on how you can run then.

### Installing the database, API and grouper-loader daemon

You will need a host with Java Development Kit installed. You will need Apache Ant for installation other components, so it's a good idea to have that set up as well. Instructions on getting your environment set up are found on the sections on the quick start. You'll probably want to ensure the the java and ant executables are on your system path, and that the JAVA_HOME and ANT_HOME system variables are set up.

The first step is to download the API from the Grouper download pages at http://www.internet2.edu/grouper/software.html at the time of writing the latest version is 1.6.2, available as a compiled binary from http://www.internet2.edu/grouper/release/1.6.2/grouper.apiBinary-1.6.2.tar.gz

On Linux, unpack the archive with tar -xzf grouper.apiBinary-1.6.2.tar.gz on Windows you will need to use a third party extraction utility, such as 7-zip from http://www.7-zip.com that we used when installing the quick start. Once you've upacked the archive, you will have a directory called grouper.apiBinary-1.6.2 - we'll be calling this GROUPER_HOME from now on.

### Setting up the database

Go to your database server and set up a database for grouper to use, together with a user name and password for grouper to use to authenticate to the database. This user will need permissions to create, modify and delete tables and views, as well as permissions to insert, modify and delete data in tables. The reason for this is that the Grouper schema is installed by Grouper itself, rather than being distributed as a set of SQL scripts. This makes it much easier to support multiple database engines, and also enables upgrade scripts to be generated.

On a Linux host with Postgres 8 installed and with the Postgres utilities in the system path I run the following commands:

```
su postgres
createdb grouperdb -E utf-8
psql
>CREATE user grouper with password 'grouper';
>GRANT ALL PRIVILEGES ON DATABASE grouperdb TO grouper;
>GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA whatever TO someuser;
\q
```

For MySQL the commands are:

```
mysql -u root (if you have set a root password in mysql you will need to use the -p switch to be prompted for a
password)
CREATE DATABASE grouperdb character set='latin1';
GRANT ALL PRIVILEGES ON grouperdb.* to 'grouper'@'localhost' IDENTIFIED BY 'grouper';
FLUSH PRIVILEGES;
```

You will need to substitute the address of the host on which you will be running the Grouper API for 'localhost' in the GRANT command for mysql. If you are unsure, you can always use 'grouper'@'*', but make sure you use a strong password if you do this. In addition, do not attempt to set the database encoding (to utf8 for example) as this will cause error complaining of keys larger than 1000 bytes when installing the database schema.

For Microsoft SQL server or Oracle, use the vendor supplied tools to create a database and set up a user will all permissions in that database.

If you are using Oracle you may be using a database that is shared with other applications, with each application having it's own schema in the database, with its name defined by the username used to log in. This can be problematic if you have to use (for some reason) a different username to log on to the database, since this can require that all tables names are qualified by the schema name in order for queries to work. This is not properly supported in the libraries that Grouper uses to connect to the database, so you may find that you need an alternative strategy for ensuring that the correct schema is used in queries. I have found myself in this situation and have had great success with a system trigger which sets the default schema on user login. This works on Oracle 10i:

```
CREATE OR REPLACE TRIGGER global_logon_trg AFTER logon ON DATABASE
DECLARE
    username varchar2(64);
BEGIN
    username:=SYS_CONTEXT('USERENV','SESSION_USER');;
    if username LIKE 'a_user' then
            EXECUTE IMMEDIATE 'ALTER SESSION SET CURRENT_SCHEMA = GROUPER';
    END IF;
END;/
```