Conventions :

In this document we will use the term **ICAT** to refer to *ICAT* itself and all of its components. Where needed we will differentiate accordingly between the components.

The different installation's tasks were divided in "*chapters*", each of them always contain three different sections
- **Important Remarks** and **What's to do** sections are self explanatory
- **What's needed** section may sometimes anticipate on tasks defined in the *What's to do* section and thus may be unclear in the first approach.

We do not intend this guide to be exhaustive enough such that you don't need to read the installation notes of the different component at all. In particular the installation notes contain valuable descriptions of the variables to be configured.

ICAT Core Installation Prerequisites

Important Remarks

ICAT is mainly written in Java (SOAP, EJB, GWT Client..) so a strong dependency on Java and on a J2EE Container is expected. It is suggested not to install the *openJDK* but instead the official *JDK 7* from *Oracle*. This will greatly simplify the installation and operation of the sofware chain.

At our facility, servers are mostly virtualised and are deployed with *Scientific Linux* (SL) as operating system (OS), currently in the version *6.4*. Furthermore we will use *Oracle 11g* as database Back-End since the Computing Division provides such a service.

What's needed

- Server Hardware or virtualised image with SL6.4 installed (or any RHEL[1] like OS).
- Oracle Java7 JDK
- Python (2.4 to 2.7)
- *Oracle 11g* as database Back-End
- Oracle *Instant Client* (11g R2)
- cx_oracle access Oracle DB using Python database API
- Oracle *Glassfish* Community Edition as J2EE Container

An installation on Windows would also be possible but will not be covered here. MySQL can also be used as database Back-End.

For a password-less SSH login to the ICAT server see the short excursion at the end of the document for a small recap on how to set it up.

Java Installation

Important Remarks

The installation of ICAT assumes the installation of a *Oracle Glassfish* J2EE container. The latest version of the ICAT fully support **Oracle Glassfish 4**[2]. Mandatory for a smooth installation and operation of Glassfish 4 is the installation of **Oracle Java7 JDK**.

What's needed

- Download and install Oracle Java 7 JDK JDK Downloads (http://www.oracle.com/technetwork/java/javase/downloads/index.html)

What's to do

```
[root@icatserver ~]# rpm -ihv jdk-7u45-linux-x64.rpm
```

Optional but evtl. mandatory: if the Java6 JDK is still needed on the same system one can use the Linux *alternatives System*. For a small recap see short excursion at the end of the document

```
[root@icatserver ~]# alternatives --install /usr/bin/java java
/usr/java/latest/bin/java 10002 --slave /usr/bin/keytool keytool
/usr/java/latest/bin/keytool --slave /usr/bin/rmiregistry rmiregistry
/usr/java/latest/bin/rmiregistry

[root@icatserver ~]# alternatives --install /usr/bin/javac javac
/usr/java/latest/bin/javac 10002 --slave /usr/bin/jar jar
/usr/java/latest/bin/jar --slave /usr/bin/rmic rmic
/usr/java/latest/bin/rmic

[root@icatserver ~]# alternatives --install /usr/bin/javaws javaws
/usr/java/latest/bin/javaws 10002 --slave /usr/bin/jcontrol jcontrol
/usr/java/latest/bin/jcontrol --slave /usr/bin/javadoc javadoc
/usr/java/latest/bin/javadoc

[root@icatserver ~]# update-alternatives --config java
```

Python Installation (Optional)

Important Remarks

ICAT extensively uses python for scripts ranging from simple connection test client up to sophisticated installation scripts.To be clear *Python* is mandatory since ICAT and all its components are installed via a `setup` command written in python. As claimed in the ICAT documentation (Component Guidelines | ICAT (http://icatproject.org/collaboration/component-guidelines/)) all installation scripts should be written in python compatible with versions 2.4 to 2.7.

Very Important: Do NOT replace/overwrite the OS installed python version otherwise it would injure the OS and bad things may happen.

There is also an alternative way of maintaining the *Python Environment_which is by using a*

Python Distribution_ such as Anaconda (http://continuum.io/downloads) and the other is Enthought Canopy (https://wwww.enthought.com/products/canopy). Using such distributions allows one to maintained its environment without breaking the system and having root privileges.

Let's assume python 2.7 should be installed, but not using such a distribution. Do not forget to update the *Library path* so that the new python libs are found by the python interpreter.

What's needed

- Prior to installing Python 2.7.6 on SL6.4 we need the *Linux Development Tools* to be installed and an updated SL6. OS.

further Optional steps: install a python package manager `pip` and python virtual environments `virtualenv`.

What's to do

```
[root@icatserver ~]# yum update
[root@icatserver ~]# yum groupinstall "Development tools"
[root@icatserver software]# wget
http://www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
[root@icatserver software]# cd Python-2.7.6
[root@icatserver Python-2.7.6]# ./configure --prefix=/usr/local --enable-
shared
[root@icatserver Python-2.7.6]# make && make altinstall
```

Very IMPORTANT : Do a `make altinstall` not a `make install`

Please add `/usr/local/lib` to the Library Path by issuing (if not already done)on may to get an `cannot open shared object` error.

```
echo '/usr/local/lib' >> /etc/ld.so.conf.d/usr_local.conf
```

and reload *Libpath* by using `ldconfig`. The filename `usr_local.conf` was arbitrarily chosen. Last but not least

```
[root@icatserver Python-2.7.6]# ln -s /usr/local/bin/python2.7
/usr/local/bin/python
```

Optional Install: if one further would like to install the python package manager `pip` and `virtualenv` follow the next steps (attention may harm your system, so be careful)

```
[root@icatserver Python-2.7.6]# wget --no-check-certificate
https://pypi.python.org/packages/source/s/setuptools/setuptools-1.4.2.tar.gz

[root@icatserver Python-2.7.6]# tar -xvf setuptools-1.4.2.tar.gz
[root@icatserver Python-2.7.6]# cd setuptools-1.4.2
[root@icatserver setuptools-1.4.2]# python2.7 setup.py install
running install
running bdist_egg
running egg_info
writing requirements to setuptools.egg-info/requires.txt
writing setuptools.egg-info/PKG-INFO
writing top-level names to setuptools.egg-info/top_level.txt
writing dependency_links to setuptools.egg-info/dependency_links.txt
writing entry points to setuptools.egg-info/entry_points.txt
reading manifest file 'setuptools.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
writing manifest file 'setuptools.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
copying setuptools.egg-info/PKG-INFO -> build/bdist.linux-x86_64/egg/EGG-
INFO
copying setuptools.egg-info/SOURCES.txt -> build/bdist.linux-x86_64/egg/EGG-
INFO
copying setuptools.egg-info/dependency_links.txt -> build/bdist.linux-
x86_64/egg/EGG-INFO
copying setuptools.egg-info/entry_points.txt -> build/bdist.linux-
x86_64/egg/EGG-INFO
copying setuptools.egg-info/requires.txt -> build/bdist.linux-
x86_64/egg/EGG-INFO
copying setuptools.egg-info/top_level.txt -> build/bdist.linux-
x86_64/egg/EGG-INFO
creating 'dist/setuptools-1.4.2-py2.7.egg' and adding 'build/bdist.linux-
x86_64/egg' to it
removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing setuptools-1.4.2-py2.7.egg
Copying setuptools-1.4.2-py2.7.egg to /usr/local/lib/python2.7/site-package

Adding setuptools 1.4.2 to easy-install.pth file
Installing easy_install script to /usr/local/bin
Installing easy_install-2.7 script to /usr/local/bin

Installed /usr/local/lib/python2.7/site-packages/setuptools-1.4.2-py2.7.egg
Processing dependencies for setuptools==1.4.2
Finished processing dependencies for setuptools==1.4.2

[root@icatserver software]# curl
https://raw.github.com/pypa/pip/master/contrib/get-pip.py | python2.7 -
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 543k 100 543k 0 0 191k 0 0:00:02 0:00:02 --:--:-- 272k
Downloading/unpacking pip
Downloading pip-1.5.tar.gz (898kB): 898kB downloaded
```

```
Running setup.py egg_info for package pip

warning: no files found matching 'pip/cacert.pem'
warning: no files found matching '*.html' under directory 'docs'
warning: no previously-included files matching '*.rst' found under directory
'docs/_build'
no previously-included directories found matching 'docs/_build/_sources'
Installing collected packages: pip
Running setup.py install for pip

warning: no files found matching 'pip/cacert.pem'
warning: no files found matching '*.html' under directory 'docs'
warning: no previously-included files matching '*.rst' found under directory
'docs/_build'
no previously-included directories found matching 'docs/_build/_sources'
Installing pip script to /usr/local/bin
Installing pip2.7 script to /usr/local/bin
Installing pip2 script to /usr/local/bin
Successfully installed pip
Cleaning up...
```

This installed `pip` the python package manager. Now install `virtualenv` using `pip`.

```
[root@icatserver software]# pip install virtualenv
Downloading/unpacking virtualenv
Downloading virtualenv-1.11.tar.gz (1.6MB): 1.6MB downloaded
Running setup.py (path:/tmp/pip_build_root/virtualenv/setup.py) egg_info for
package virtualenv

warning: no previously-included files matching '*' found under directory
'docs/_templates'
warning: no previously-included files matching '*' found under directory
'docs/_build'
Installing collected packages: virtualenv
Running setup.py install for virtualenv

warning: no previously-included files matching '*' found under directory
'docs/_templates'
warning: no previously-included files matching '*' found under directory
'docs/_build'
Installing virtualenv script to /usr/local/bin
Installing virtualenv-2.7 script to /usr/local/bin
Successfully installed virtualenv
Cleaning up...
```

Oracle Core Installation

Important Remarks

Only one *Oracle* schema will be created to hold **all** database objects created by the different ICAT components.

What's needed

- Create an *Oracle* Schema (either by asking a DBA or by creating it yourself ;))
- The Oracle *Instant Client 12.1* (Linux) will provide the libraries and tools needed to communicate with the Oracle Back-End.
- For Java, the JDBC Drivers has to be downloaded.

Go to OTN[3] Website and download the Instant Client for Linux x86_64 as *RPM*

These files are needed

- oracle-instantclient12.1-basic–12.1.0.1.0–1.x86_64.rpm
- oracle-instantclient12.1-odbc–12.1.0.1.0–1.x86_64.rpm
- oracle-instantclient12.1-sqlplus–12.1.0.1.0–1.x86_64.rpm

Optional: create a `tnsnames.ora` to be used with the `sqlplus` console utility (test connectivity)

What's to do

```
-- creates a schema named ICAT with password and roles
-- USER SQL
CREATE USER "ICAT" IDENTIFIED BY "YourPasswordHere"
DEFAULT TABLESPACE "ICAT_DATA"
TEMPORARY TABLESPACE "TEMP"
ACCOUNT UNLOCK ;

-- ROLES
Grant CONNECT to icat;
Grant DEVELOPER to icat;

ALTER USER "ICAT" DEFAULT ROLE "CONNECT","DEVELOPER";

-- SYSTEM PRIVILEGES

-- QUOTAS
ALTER USER "ICAT" QUOTA UNLIMITED ON ICAT_DATA;
ALTER USER "ICAT" QUOTA UNLIMITED ON ICAT_INDX;
```

where the Oralce role `DEVELOPER` embraces all the needed permissions a developer may need.

Install the RPMs

```
[root@icatserver software]# rpm -ihv oracle-instantclinet12.1-*
```

Optional : Test the connection with the Oracle schema ICAT using *SQL*Plus*. Remember SQL*Plus needs a `tnsnames.ora` file to resolve the service name.

- Where to found `sqlplus` : `/usr/lib/oracle/12.1/client64/bin/sqlplus`
- Where to found/create `tnsnames.ora` : `export TNS_ADMIN=/usr/lib/oracle/12.1/client64/lib/`

The execution of `sqlplus` may fail because the LD_Library Path was not updated. Thus update it

```
[root@icatserver etc]# echo "/usr/lib/oracle/12.1/client64/lib/" >
/etc/ld.so.conf.d/oracle_libs.conf
```

now add, update or create an entry in the `tnsnames.ora` file.

```
PRODDB =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL=TCP)(HOST=myhost.facility.ext)(PORT=1521))
)
(CONNECT_DATA =
(SERVICE_NAME=proddb.internal.facility.ext)
)
)

[root@icatserver software]# export
TNS_ADMIN=/usr/lib/oracle/12.1/client64/lib && sqlplus icat@proddb.
```

Oracle Python Lib Installation

Important Remarks

sources: ICAT components install notes

This installation is needed because the ICAT python installation scripts need to connect to Oracle (ex: for populating the ICAT schema with tables)

What’s needed

Download `cx_oracle` from here (http://cx-oracle.sourceforge.net/), a python Library for accessing Oracle conforming to the Python Database API

- take the `CentOS 5 x86_64 RPM (Oracle 11g, Python 2.7)` RPM.

What’s to do

install it using

```
[root@icatserver software]# rpm -ihv cx_Oracle-5.1.2-11g-py27-1.x86_64.rpm
```

Oracle JDBC driver Installation

Important Remarks

This installation is also needed because many ICAT components are written in Java and need a connection to Oracle.

What's needed

- download `ojdbc6.jar` from here : Oracle Database 11g Release 2 JDBC Driver Downloads (http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html?ssSourceSiteId=otnjp)

What's to do

copy `ojdbc6.jar` either to `<glassfish_domain_dir>/lib`, for local domain deployment, or to `/home/glassadmin/glassfish4/glassfish/lib` for a glassfish wide deployment. See below `<glassfish_domain_dir>` definition.

Glassfish Container Installation

Important Remarks

sources : New ICAT install note (http://icatproject.org/installation/glassfish/), internal knowledge

The Oracle Glassfish Platform exists in two variants as *Glassfish* or as *OGS*. OGS[4] is the commercial version whereas Glassfish is the Open Source edition. Use Glassfish 4 OE and download the ZIP file, avoid installer version.

Some local conventions

It's assumed (not mandatory) that the **Glassfish** software is installed in the *glassadmin HOME* folder (see below) and that the current installation hosts a single instance with the standard 8181 and 4848 ports. If the *Glassfish* server should run multiple domains then be cautious when assigning port number to the listener.

glassfish_domain_dir

Glassfish ICAT directory is `/home/glassadmin/glassfish4/glassfish/domains/icat-domain1`.

jre_sec_dir

directory where the Java *keyStore/trustStore* is located (`cacerts` or `jssecacerts`)

What's needed

- a dedicated user account should be designate for running Glassfish since executing Glassfish server processes as `root` is not recommended.
- download `glassfish-4.0.zip` from GlassFish Server - Download Page (https://glassfish.java.net/download.html)
- recreate a Glassfish ICAT Domain and configure its security

What's to do

Add User

```
[root@icatserver ~]# useradd --create-home -c "Glassfish Administrator" glassadmin
```

Furthermore ensure that `hostname -f` really returns the full qualified hostname.

For simplicity assume that *Glassfish* will be started from `glassadmin` user account. Login as `glassadmin`

```
[root@icatserver ~]# su - glassadmin
```

unpack Glassfish ZIP in user *$HOME* add `bin` to user *$PATH*

```
[glassadmin@icatserver ~]$ export PATH=~/glassfish4/glassfish/bin:$PATH
```

Glassfish can be either administrated using a GUI[5] or using the `asadmin` Glassfish administration utility (an executable script in the bin dir).

Important The `asadmin` command can also be executed remotely using the `--host icatserver.facility.ext` asadmin's option. see here (http://docs.oracle.com/cd/E18930_01/html/821-2416/giobi.html#scrolltoc) for more details.

Let's reconfigure Glassfish from scratch (as done in the ICAT installation), thus removing the default domain `domain1` (for readibility reason the prompt was removed). When creating a new domain a admin username/password will be asked, do not use an empty password. (GUI Admin Console is available under `https://icatserver:4848`)

```
asadmin delete-domain domain1
asadmin create-domain icat-domain1

asadmin start-domain icat-domain1
asadmin enable-secure-admin

# You must restart all running servers for the change in secure admin to
take effect.
asadmin stop-domain
asadmin start-domain icat-domain1

# One can now seccurely access (also remotly) the site with the certificate
which was self signed and generated with the `create-domain` option.
# this will save admin password locally (password-less login)
asadmin login

# Set log format `%client.name% %auth-user-name% %datetime% %request%
%status% %response.length%`
asadmin set server.http-service.access-log.format="common"
asadmin set server.http-service.access-logging-enabled=true
asadmin set server.thread-pools.thread-pool.http-thread-pool.max-thread-
pool-size=128

# If GUI : Configurations > server-config > HTTP Service > Http Listeners
asadmin delete-ssl --type http-listener http-listener-2
asadmin delete-network-listener http-listener-2

# Create a network listener with different SSL Support as default
asadmin create-network-listener --listenerport 8181 --protocol http-
listener-2 http-listener-2
asadmin create-ssl --type http-listener --certname s1as --ssl3enabled=false
--ssl3tlsciphers +SSL_RSA_WITH_RC4_128_MD5,+SSL_RSA_WITH_RC4_128_SHA http-
listener-2

# Mandatory to enable to run ICAT on Glassfish4
asadmin set configs.config.server-config.cdi-service.enable-implicit-
cdi=false
```

Glassfish Security Configuration

Important Remarks

When manipulating files related to Glassfish security first **shutdown** Glassfish. This in particular applies to SSL Certificates.

Keystores:

Glassfish by default is pre-configured with two SSL certificates contained in the keystore `<glassfish_domain_dir>/config/keystore.jks` and has two *aliases* defined `s1as` and `glassfish-instance`. This can be shown with

```
[glassadmin@pandacats config]$ keytool --list -keystore keystore.jks --
storepass changeit

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 2 entries

glassfish-instance, Jan 20, 2014, PrivateKeyEntry,
Certificate fingerprint (SHA1):
2C:25:83:05:85:F5:3C:1E:C4:E0:06:D3:73:DF:0C:87:16:4A:F8:45
s1as, Jan 20, 2014, PrivateKeyEntry,
Certificate fingerprint (SHA1):
F9:4B:99:82:68:FC:92:F1:1E:EF:D0:BB:D2:00:83:0B:61:3B:83:32
```

here the default *master password* `changeit` is used. Please be aware that everybody knows the Keystore `keystore.jks` and the master master password, so anyone may be able to capture and read the data sent over the SSL connection (*https*). Therefore new certificates should be generated for the aliases **s1as** and **glassfish-instance** and should be imported into one of the JRE SSL system-wide **trustStores** `cacerts` or `jssecacerts`. Remember that `jessecacerts` has higher priority than `cacerts`, both located in <jre_sec_dir>. `jssecacerts` is missing by default.

The procedure described in Glassfish | ICAT (http://icatproject.org/installation/glassfish/) will only extract the `s1as` certificate and import it into the trustStore `jssecacrets` (in <jre_sec_dir>) without generating a new certificate . So be aware of that! (using the procedure described by the developers. See Short excursions for more details)

The store `<glassfish_domain_dir>/config/cacerts.jks` is Glassfish *trustStore* containing the (public) certificates from known (trusted) Certificate Authority.

Passwords:

Two separate passwords are used and maintained by Glassfish to secure its infrastructure. Both passwords can be manipulated using the `aasadmin` utility.
- master password : protects domain-encrypted files such as the Glassfish Keystore `keystore.jks`. Default value is `changeit`
- administration password : the password of the Glassfish administrator (used in `asadmin login`)

Changing the master password may raise some errors so be cautious and first backup the following files located in <glassfish_domain_dir>

- domain-passwords
- keystore.jks
- cacerts.jks

for more details see the *Short Excursion* chapter.

What's needed

- Extract SSL Certificate with the alias `s1as` from Glassfish keyStore and import into the Java7 JRE trustStore.

Remember : certificate with alias `s1as` is the default alias of the Glassfish server!!

What's to do

Go to `<glassfish_domain_dir>/config`

```
[glassadmin@icatserver config] keytool -exportcert -keystore keystore.jks -
file GF-s1as-export.cert -storepass changeit -alias s1as
```

Check if ok using

```
[glasshadmin@icatserver config]$ keytool -printcert -file GF-s1as-
export.cert
Owner: CN=icatserver.facility.ext, OU=GlassFish, O=Oracle Corporation,
L=Santa Clara, ST=California, C=US
Issuer: CN=icatserver.facility.ext, OU=GlassFish, O=Oracle Corporation,
L=Santa Clara, ST=California, C=US
Serial number: 9e8dwf2
Valid from: Tue Jan 14 17:43:40 CET 2014 until: Fri Jan 12 17:43:40 CET 2024

Certificate fingerprints:
MD5: 58:42:6E:DD:E0:FC:76:6F:CB:D0:57:61:48:FF:27:BA
SHA1: BD:AA:FC:71:22:0E:58:E3:44:75:E4:CD:C5:9F:55:13:F6:D4:B2:33
SHA256:
A1:5B:7B:20:50:24:6A:E9:2B:B6:AC:68:83:41:E3:E1:53:83:80:7D:4E:21:09:2C:CA:97
:55:B0:AA:EE:F2:6D
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 5F 02 A8 E9 FF 61 9E FE 86 FF EA EE 0D E6 14 7D _....a.........
0010: C2 6F E2 D2 .o..
]
]
```

Import now `s1as` certificates into JRE default trustStore `<jre_sec_dir>/jssecacerts`. This must be executed as `root`. By default you only have a `cacerts` store no `jssecacerts` so be careful not to overwrite if it already exists.

```
[root@icatserver security]# cp cacerts jssecacerts
[root@icatserver security]# keytool -importcert -keystore jssecacerts -file
/home/glassadmin/glassfish4/glassfish/domains/icat-domain1/config/GF-s1as-
export.cert -storepass changeit -alias `hostname -f` -noprompt
Certificate was added to keystore
```

Troubleshooting SSL errors

If an error arises one could try SSLPoke from the company `Atlassian`. Let's take at a common error when the keystore file is not correctly spelled, or you don't remember the name of the keystore (so we must test candidat store files). One could also use the `keytool` utility.

Test SSL connection to icatserver

```
[glassadmin@icatserver ~]$ java SSLPoke icatserver 8181
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:385)
at
sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:292)
at sun.security.validator.Validator.validate(Validator.java:260)
at
sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:326
)
at
sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:
231)
at
sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImp
l.java:126)
at
sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:13
23)
at
sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:153)
at sun.security.ssl.Handshaker.processLoop(Handshaker.java:868)
at sun.security.ssl.Handshaker.process_record(Handshaker.java:804)
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1016)
at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:13
12)
at sun.security.ssl.SSLSocketImpl.writeRecord(SSLSocketImpl.java:702)
at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:122)
at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:136)
at SSLPoke.main(SSLPoke.java:31)
Caused by: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBui
lder.java:196)
at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:268)
at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:380)
... 15 more
```

Test a SSL connection for known working service like Mail over SSL

```
[glassadmin@icatserver ~]$ java SSLPoke mail.facility.ext 443
Successfully connected
```

It seems the java JRE cannot find the correct trusted keystore. Thus try to specify it on the command line as

```
[glassadmin@icatserver ~]$ java -
Djavax.net.ssl.trustStore=/usr/java/latest/jre/lib/security/wrong-
jssecacerts SSLPoke icatserver 8181
Successfully connected
```

So the keystore was misspelled and should be renamed to `jssecacerts`.

Last example also the wrong keyStore, here `s1as` certificate is missing

```
[glassadmin@icatserver ~]$ java -
Djavax.net.ssl.trustStore=/usr/java/latest/jre/lib/security/cacerts SSLPoke
icatserver 8181
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target
at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:385)
at
sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:292)
at sun.security.validator.Validator.validate(Validator.java:260)
at
sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:326
)
at
sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:
231)
at
sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImp
l.java:126)
at
sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:13
23)
at
sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:153)
at sun.security.ssl.Handshaker.processLoop(Handshaker.java:868)
at sun.security.ssl.Handshaker.process_record(Handshaker.java:804)
at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1016)
at
sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:13
12)
at sun.security.ssl.SSLSocketImpl.writeRecord(SSLSocketImpl.java:702)
at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:122)
at sun.security.ssl.AppOutputStream.write(AppOutputStream.java:136)
at SSLPoke.main(SSLPoke.java:31)
Caused by: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target
at
sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBui
lder.java:196)
at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:268)
at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:380)
... 15 more
```

ICAT Software Installation

Sources : Current Releases artefacts (http://icatproject.org/releases/current-release/)ICAT Components | ICAT (http://icatproject.org/installation/component/), perhaps also important ICAT server EAR (http://icatproject.org/mvn/site/icat/4.3.2/icat.ear/installation.html#installation).

Important General Remarks

The setup procedure is quite similar to installing software on *NIX system using the `make` utility, namely `configure && install`.

```
[glassadmin@icatserver dir-ICAT-Comp] ./setup --verbose configure
[glassadmin@icatserver dir-ICAT-Comp] ./setup --verbose install
```

To display the help use the `-h` option, and to uninstall simply issue in the unpacked distribution folder `./setup uninstall`.

For details on the common installation procedure see ICAT component | ICAT (http://icatproject.org/installation/component/).

In general most of the ICAT Components will come with at least two *java properties* files used to configure the installation process and the software itself. Per default those files have the extension `.example`. The *configure* step copies the *example* files to their *properties* counterparts ready for editing. One could also copy all `*.properties.example` to their *non example* counterparts directly.

The `<compo>-setup.properties` is used to configure the installation whereas the `<compo>.properties` will configure the software and will be copied into the `<glassfish_domain_dir>/config` folder.

The properties files must change their *NIX permissions to **600** (otherwise the *configure* step will complain *'.properties' must not be world readable*)

ICAT allows different *authentication mechanisms* to authenticate a user and each of them has its own versioning as well as its own maintenance frequency. The *Current Releases artefacts* link above will give a list of the latest versions. ICAT is highly modular and thus each setup may be different.

Currently four ICAT Authentication plugins are available.

- **authn_anon** anonymous access
- **authn_simple** simple authentication using a Java properties file of the server

- **authn_db** uses a table-based approach (username,password)
- **authn_ldap** well known LDAP server authentication.

The *ICAT* installation seems[6] to be sensible on the order in which the installation is performed. Therefore it is strongly suggested to install the needed authentication plugins **prior** to install **ICAT Core** itself.

if you have a Oracle DB Password containing symbols then use

```
password="'"A/complicated*Pass$word?"'"
```

in JDBC DB Connection String (see later)

What's needed

- In this setup only`authn_ldap`, `authn_simple` and `authn_db` will be used as authentication mechanism.
- The following directories must be created before launching the installation scripts
  - ../bin (icat commands) , ../java (icat-setup) , ../logs, ../data (lucene)
- Download ICAT components

What's to do

Make the folders described above. Download the components.

Sample output from a download

```
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/icat.ear/4.3.2/icat.ear-
4.3.2-distro.zip
--2014-01-16 16:52:53--
http://icatproject.org/mvn/repo/org/icatproject/icat.ear/4.3.2/icat.ear-
4.3.2-distro.zip
Resolving icatproject.org... 130.246.143.9
Connecting to icatproject.org|130.246.143.9|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13706276 (13M) [application/zip]
Saving to: “icat.ear-4.3.2-distro.zip”
.......
2014-01-16 16:52:54 (7.06 MB/s) - “icat.ear-4.3.2-distro.zip” saved
[13706276/13706276]
```

Now the rest of the components to be installed

```
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_anon.ear/1.0.1/authn_a
non.ear-1.0.1-config.zip
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_anon.ear/1.0.1/authn_a
non.ear-1.0.1.ear
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_db/1.1.1/authn_db-
1.1.1-distro.zip
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_ldap/1.1.0/authn_ldap-
1.1.0-distro.zip
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_simple.ear/1.0.0/authn
_simple.ear-1.0.0-config.zip
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/authn_simple.ear/1.0.0/authn
_simple.ear-1.0.0.ear
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/icat.client/4.3.2/icat.clien
t-4.3.2.jar
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/icat-setup/1.1.0/icat-setup-
1.1.0-distro.zip
[glassadmin@icatserver ICAT]$ wget
http://icatproject.org/mvn/repo/org/icatproject/ice/1.0.0/ice-1.0.0-
distro.zip
```

ICAT Authentication Plugins Installation

Authentication plugins provide different mechanisms to identify a user

Important Remarks

Authentication plugins must be installed **before** any other ICAT component.

The `authn_db` and `autn_simple` authenticator plugins need to be provided the username/password **manually** either in table or in a file respectively. The password may be entered in clear text without a leading `$` or as a cryptographic hash value of the password. Thus if one provides an hash value then itmust start with a '$' and must be provided in the same format as in the shadow password file (shadow(5))

To generate such password use `mkpasswd` (debian like OS) or `grub-crypt` (redhat like OS)

```
[glassadmin@icatserver ~]$ grub-crypt --sha-512
Password:
Retype password:
$6$WyjzXHWAajMN9Nfd$roFbDGMyhKgKWUFxT8.F9S9jrkUOG8tBY4s2ndnYG6QDNaIrqiMbWNy/
0Gcz8.FMhwIkbp.3p.chf2ipqw.8T.
```

or

```
[glassadmin@icatserver ~]$ mkpasswd --method=sha512 --salt=vb1tLY1qiY
PASSWORD
```

here we only consider the **sha512** hashing algorithm and AFAIK `grub-crypt` has no option to provide a salt value.

What's needed

- *authn_simple* installation note ICAT SIMPLE Authn EAR - Authn SIMPLE Plugin Installation (http://icatproject.org/mvn/site/authn_simple/1.0.0/authn_simple.ear/installation.html)
- *authn_db* installation note ICAT DB Authn - Authn DB Plugin Installation (http://icatproject.org/mvn/site/authn_db/1.1.1/installation.html)
- *authn_ldap* installation note ICAT LDAP Authn - Authn LDAP Plugin Installation (http://icatproject.org/mvn/site/authn_ldap/1.1.0/installation.html)
- ICAT software is already downloaded

What's to do

see the Sub-Chapters below.

Authentication Plugin : authn_simple

- Unpack `authn_simple.ear-1.0.0-config.zip` and configure it by editing the properties file `authn_simple.properties`.

- Define the users in the property `user.list` and their associated passwords `user.<user>.password` where non clear text password starts with `$`.

- Deploy the `EAR` using `asadmin` utility.

```
[glassadmin@icatserver authn_simple]$ chmod 600 authn_simple.properties
[glassadmin@icatserver authn_simple]$ cp authn_simple.properties
<glassfish_domain_dir>/config
[glassadmin@icatserver authn_simple]$ asadmin --port 4848 deploy
authn_simple.ear-1.0.0.ear
Application deployed with name authn_simple.ear-1.0.0.
Command deploy executed successfully.
```

- check `server.log` or `authn_simple.log` for failure. Both logs in `<glassfish_domain_dir>/logs`.

Authentication Plugin : authn_db

- Unpack `authn_db-1.1.1-distro.zip` and do a `setup configure`. The *configure* step will check and prepare the configuration file and prompt you to edit the newly created *authn_dbsetup.properties* file.

```
[glassadmin@icatserver authn_db]$ ./setup --verbose configure
```

Uncomment the *Oracle* section, comment the *MySQL* section out and point the `glassfish` property to the appropriate directory.

```
#Oracle Database Properties
driver=oracle.jdbc.pool.OracleDataSource
dbProperties=url="'"jdbc:oracle:thin:@//myhost:1521/proddb.internal.facility
.ext"'":ImplicitCachingEnabled=true:MaxStatements=200:user=icat:password=You
rPassWordHere
```

If your password contains symbols then it need to be enclosed in quotes. See *Short Excursions* at the end of the document for more details and hints.

_ Save and change file permission (`chmod 600`). Then install

```
[glassadmin@icatserver authn_db]$ ./setup --verbose install

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
property.administrative.domain.name
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
resource jdbc/authn_db
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
connection-pool authn_db
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
connection-pool --datasourceclassname oracle.jdbc.pool.OracleDataSource --
restype javax.sql.DataSource --failconnection=true --steadypoolsize 2 --
maxpoolsize 8 --ping --property
url="'"jdbc:oracle:thin:@//myhost:1521/proddb.internal.facility.ext"'":Impli
citCachingEnabled=true:MaxStatements=200:user=icat:password=YourPasswordHere
authn_db
JDBC connection pool authn_db created successfully.
Attempting to ping during JDBC Connection Pool Creation : authn_db -
Succeeded.
Command create-jdbc-connection-pool executed successfully.

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
resource --connectionpoolid authn_db jdbc/authn_db
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy
authn_db-1.1.1.war
```

The install procedure creates only **one table** named `PASSWD` in the Oracle *ICAT* Schema. Edit the table by filling some username and password. Same password policy applies here as with *auth_simple* . Check `authn_db.log` and `server.log` in `<glassfish_domain_dir>/logs/` for errors.

Next a SQL statement for inserting users

```
insert into PASSWD (username,encodedpassword) values ('guest_star',
'aStrong_password')
insert into PASSWD (username,encodedpassword) values ('user1',
'$5$rOcFOsg2dEk5$fB15Q6sbW0TAAlngk.Hmciuep59G')
```

Authentication Plugin : authn_ldap

Important Remarks

The *authn_ldap* plugins is easy to install but its usage is not. Some *LDAP* know-how might be very helpful to configure this plugin. The plugin's documentation is not really exhaustive about its configuration. See below and the *Testing the Authentication Plugins….* Section for details.

One needs to understand how to configure the properties file `authn_ldap.properties` in order to operate this plugin. The two mandatory properties are

1. `provider_url`
2. `security_principal`

The `provider_url` accepts LDAP URLs. Remember `ldap://` protocol will transmit password in clear text whereas `ldaps://` will encrypt the connection.

```
- ldaps://ad.facility.ext
- ldaps://ad.facility.ext/ou=Users,dc=ad,dc=facility,dc=ext
```

For the `security_principal`, as stated in the `authn_ldap.properties`, one should use the wildcard character `%` in place of the username to authenticate.

Examples of `security_principal`, some examples are restricted to a Microsoft Active Directory Environment (WinAD). By restricted it is meant that the authors tested it against such an environment (perhaps it's also work in other LDAP Env).

```
- security_principal % (ALL, digest md5))
- security_principal <win_domaim_name>\\% (WinAD)
- security_principal %@<AD_kerberos_realm> (WinAD)
- security_principal <userPrincipalName > (WinAD)
- security_principal CN=%,ou=Users,ou=Facility,dc=ad,dc=facility,dc=ext
(DN,WinAD)
- security_principal uid=%,ou=Users,ou=Facility,dc=ad,dc=facility,dc=ext
(DN,OpenLDAP)
```

where

- <win_domain_name> = a Windows Domain Name (like the well known Microsoft CONTOSO example Domain Name)
- <AD_kerberos_realm> = Active Directory Keberos Realm Name (here only tested against Active directory (AD) kerberos)
- sometimes in AD environment it is allowed to login using the `userPrincipalName` which may be set alike the email address. In this case the could be `%@facility.ext`.

The DN approaches may be used to restrict the authentication to dedicated subtrees. But the eventuality should be considered that, not all allowed users objects (principals) leave in the same subtree.

By default the *LDAP Context* will use **simple** authentication method (username/password auth) and will use a **SUBTREE_SCOPE** search method, i.e. descend the tree until the principal is found. Thus only defining `%` as `security_principal` will start a recursive search at the **root node** of the DIT and thus can be time consuming, depending on the DIT size and complexity. Of course such a setting offers also the most flexibility to authenticate users against multiple branches since one may use the DN form in the username (see later). But this also implies that the user is aware of the ICAT internals and this is of course not wanted.

Sometimes a (authenticated) username is known as the *principal* and the password as the *credential*.

What's to do

- Unpack `authn_ldap-1.1.0-distro.zip` and edit the `authn_ldap-setup.properties` *properties* files.

- execute the first configure step

```
[glassadmin@icatserver authn_ldap]$ ./setup --verbose configure
```

- For the moment make sure that only the attributes `provider_url` and `security_principal` are uncommented while the others are commented out.

```
[glassadmin@icatserver authn_ldap]$ ./setup --verbose install

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
property.administrative.domain.name
authn_ldap.properties copied to
/home/glassadmin/glassfish4/glassfish/domains/icat-domain1/config
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy
authn_ldap-1.1.0.war
```

All authentication plugins are now installed and one should **check** that **all** authentication plugins work as expected. For this purpose a handy **test script** exists but needs the deployment of the *ICAT Core* component

Testing the Authentication Plugins and the authn_ldap Configuration

As stated above once *ICAT Core* is installed, a test script will be available `testicat` :

```
testicat https://icatserver:8181 <auth mnemonic> username <name> password
<pw>
```

where

- `<auth mnemonic>` is the mnemonic to define the authentication method to use here `db,ldap,simple`
- `<name>` is the username of the user to authenticate (more globally could be entered in one the *forms* described earlier for the security principal. The greatest flexibility is achieved when the `security_principal` is defined as `%` and the `<name>` controls the subtree to be searched.
- `<pw>` is clear ;-)

The `testicat` command authenticates and performs a dry-run with some CRUD[7] operations. It will raise an error (Warning) if the user is not declared in the `rootUserNames` property (defined in `icat.properties`) since only those users have full CRUD permissions (see later)

next some examples on how to authenticate users

```
[glassadmin@icatserver authn_ldap]$ testicat https://icatserver:8181 simple
username userA password password_in_clear_text
[glassadmin@icatserver authn_ldap]$ testicat https://icatserver:8181 simple
username userA password -
[glassadmin@icatserver authn_ldap]$ testicat https://icatserver:8181 db
username userDB_1 password -
[glassadmin@icatserver authn_ldap]$ testicat https://icatserver:8181 ldap
username AD_user password -
```

Remember that using a *dash* in place of a password will prompt for a password (not echoed). Now a typical `testicat` output, with user having no permissions but passing the authentication successfully

```
[glassadmin@icatserver config]$ testicat https://icatserver:8181 simple
username userA password password_in_clear_text
Logged in as userA with 119.998333333 minutes to go
Traceback (most recent call last):
File "/home/fishadmin/bin/testicat", line 46, in <module>
service.create(sessionId, group)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 542, in
__call__
return client.invoke(args, kwargs)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 602, in
invoke
result = self.send(soapenv)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 649, in
send
result = self.failed(binding, e)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 702, in
failed
r, p = binding.get_fault(reply)
File "/usr/local/lib/python2.7/site-packages/suds/bindings/binding.py", line
265, in get_fault
raise WebFault(p, faultroot)
suds.WebFault: Server raised fault: 'CREATE access to this Grouping is not
allowed.'
```

Next the output of a user defined in the `rootUserNames` and a dash as password

```
Password:
Logged in as adminUserLDAP with 119.998666667 minutes to go
Login, search, create, delete and logout operations were all successful.
```

** OPTIONAL**

To illustrate the "complexity" of the `authn_ldap`'s configuration consider the following output

```
Password:
Traceback (most recent call last):
File "/home/fishadmin/bin/testicat", line 35, in <module>
sessionId = service.login(plugin, credentials)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 542, in
__call__
return client.invoke(args, kwargs)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 602, in
invoke
result = self.send(soapenv)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 649, in
send
result = self.failed(binding, e)
File "/usr/local/lib/python2.7/site-packages/suds/client.py", line 702, in
failed
r, p = binding.get_fault(reply)
File "/usr/local/lib/python2.7/site-packages/suds/bindings/binding.py", line
265, in get_fault
raise WebFault(p, faultroot)
suds.WebFault: Server raised fault: 'The username and password do not match'
```

here a wrong password was provided and thus error message would be ok, **BUT** the same error message will be displayed if the plugin's configuration was incorrectly made. By *incorrect* it is meant errors ranging from simple misspellings to providing wrong kind of arguments (see below for example).

A further source of error is the, what I called, *implicit LDAP Search error*.

The implicit LDAP Search

The plugin offers a way to map a user name onto a new name. This done using the properties `ldap.base, ldap.filter or ldap.attribute` and they all must be defined at the same time or none. A property is *defined* when it is **uncommmented** and has a value assigned to it, and at contrary is *undefined* when **commented out**.

Indeed if the properties described above are defined in `authn_ldap.properties` then the `authn_ldap` plugin will initialize an (implicit) *LDAP Context Search* using the values defined in those properties. This of course allows for new kinds of error.

- wrong or misspelled `ldap.filter` or `ldap.base`
- user to authenticate not found using the given base DN and filter
- etc..

As stated before one must know its LDAP infrastructure (and in particular the DIT[8] structure) before using such a search facility (the same applies for DN forms of the principal)

BUT a more critical issue is that at our facility (running a Microsoft AD Environment) such an LDAP search can only be performed by a **manager account**. Thus if the principal is not allowed to perform such searches then the authentication will fail BUT the error message raised is still *suds.WebFault: Server raised fault: 'The username and password do not match'*.

So if the users are not allowed to perform LDAP searches then simply let all three properties undefined (commented out).

Two *NIX commands exist to test the setting

```
ldapsearch -H ldaps://<ldap_server> -W -x -D CN=<manager
account>,OU=Services,OU=IT,DC=ad,DC=facility,DC=ext -b
ou=Users,ou=Facility,dc=ad,dc=facility,dc=ext -LLL -E pr=1000/noprompt "(&
(objectCategory=Person)(sAMAccountName=*)(initials=*)(employeeID=*)(cn=
<usernname>))"

ldapwhoami -H ldaps://<ldap_server> -W -x -D
"CN=testuser.111@facility.ext,ou=Project
Users,ou=Users,ou=Facility,dc=ad,dc=facility,dc=ext"
```

ICAT core installation

ICAT Core provides the cataloging of data using metadata. It provides really core functionalities, no GUI, no concept of user login, permissions or whatever. In particular the installation will create most of the tables used by the different ICAT Components.

Important Remarks

As stated above make sure you installed all the needed authentication plugins before installing *ICAT core*. The order in which the installation is performed somehow really matter !!

ICAT core comes with three different `properties` files, `icat-setup.properties, icat.properties, icat.log4j.properties`. Be really cautious when editing the `icat.properties` file since the installation scripts do not always raise adequate error messages. Pay also attention to the fact that the installation will not stop if an error is encountered. For those among you looking for some debugging help see the debugging section at the end of the document.

When you execute `./setup configure` you may see the following messages

```
[glassadmin@icatserver icat]$ ./setup --verbose configure

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
property.administrative.domain.name
You are using Glassfish version 4

Value for authn.list in icat.properties is 'db simple' which differs from
example: 'db ldap anon simple'
Value for rootUserNames in icat.properties is 'guest1 guest2 admin' which
differs from example: 'root'
```

These messages can be ignored since they are only warnings and actually informing us that the properties files were edited.

After a successful `configure` a file named `configured` will be written to the *icat* directory.

ICAT Core will install a script `testicat` which can be used to test the connectivity of the installation.

What's needed

- ICAT Core Installation Note ICAT Server ".ear" - ICAT Server Installation (http://icatproject.org/mvn/site/icat/4.3.2/icat.ear/installation.html)
- Oracle Install Client 12.1 and cx_oracle is installed and working.
- `icat.ear-4.3.2-distro.zip` should already be downloaded.

What's to do

- Unpack `icat.ear-4.3.2-distro.zip` in user *$HOME* folder.
- Edit `icat-setup.properties` and `icat.properties`
    - In `icat-setup.properties` uncomment the Oracle Section and change JDBC Connection URL and point the `glassfish` property to the correct folder
    - In `icat.properties`
        - change the login names defined in `rootUserNames`
        - list the installed authenticator plugins in `authn.list`
        - enable / disable each JNDI name according to the authenticators listed in the `authn.list` property. Thus if an authenticator plugin is not installed then do not list it in `authn.list` and comment its corresponding *JNDI* Name.
    - check the rest of the properties and try to get an understanding of them by reading the installation note ;-)
- Now install the software

```
[glassadmin@icatserver icat]$ ./setup --verbose install

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
property.administrative.domain.name
You are using Glassfish version 4

icat.properties copied to
/home/glassadmin/glassfish4/glassfish/domains/icat-domain1/config
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
resource jdbc/icat
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
connection-pool icat
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jms-
resource jms/ICAT/TopicConnectionFactory
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jms-
resource jms/ICAT/Topic
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
connection-pool --datasourceclassname oracle.jdbc.pool.OracleDataSource --
restype javax.sql.DataSource --failconnection=true --steadypoolsize 2 --
maxpoolsize 8 --ping --property
url="'"jdbc:oracle:thin:@//myhost:1521/proddb.internal.facility.ext"'":Impli
citCachingEnabled=true:MaxStatements=200:user=icat:password=YourPasswordHere
icat
JDBC connection pool icat created successfully.
Attempting to ping during JDBC Connection Pool Creation : icat - Succeeded.
Command create-jdbc-connection-pool executed successfully.
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
resource --connectionpoolid icat jdbc/icat
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jms-
resource --restype javax.jms.TopicConnectionFactory
jms/ICAT/TopicConnectionFactory
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jms-
resource --restype javax.jms.Topic jms/ICAT/Topic
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy --
deploymentorder 100 icat.ear-4.3.2.ear

testicat copied to /home/glassadmin/bin

icatadmin copied to /home/glassadmin/bin
```

testicat is a “simple” (SOAP) client to test the connectivity to the *Icat CORE* server.

```
testicat https://icatserver:8181 db username guest_star password
aStrong_password
```

where

- db: the authenticator mechanism to be used (use mnemonic)
- username and password are keywords for identifying the kind of credential value passed

..

- guest_star, aStrong_password are the values to be used as credential.

Remark : testicat needs all arguments otherwise errors are raised. Many *ICAT* CLI utilities need to be provided some user credentials and this should be done always the same way namely as `username aUser password aPasswd` (always provided all four arguments (keywords and values)). An important feature is the possibility (and is also strongly recommended to use it when appropriate) to specify the password value as `–` (a Dash). By doing so the utility should prompt for the password without echoing it. Not suitable when used with a script.

Component Installation : ‘icat-setup’

icat-setup provides the software component to manage the users, groups and their permissions (named rules). This is a command line only utility. This component adds a permission layer to *ICAT*. Actually Icat Core already provides silently the persistent layer for the permission model managed by `icat-setup`. Indeed the deployment of the *Icat EAR* in the previous step also creates most of the tables the *ICAT* system will use. An exception is for instance *authn_db* that will create its own `PASSWD` table. By default the complete permission layer is **empty** i.e. the tables are empty. This behavior is indeed only an hypothesis that I founded out by doing some reverse engineering.

Import Remarks

Only the **admin** users listed in the Java property `rootUserNames` of the `icat.properties` file are allowed to use `icat-setup`. The location of the file is `<glassfish_domain_dir>/config`. Otherwise the original file is located in `/home/glassadmin/icat/`

Remember rule of thumb of ICAT, most if not all **icat** commands only accept an even number of arguments when called not interactively. For instance calling the `icat-setup` statement from above with the *listuser* as further argument will fail as shown next.

```
[glassadmin@icatserver ~]$ icat-setup https://icatserver.facility.ext:8181
db username admin password adm_secure_password listuser
Credential list must of even length with a series of name value e.g.
'username root password secret'
```

`icat-setup` can only be run interactively or from a file by providing `-f` option. See later for some explanation.

Very Important As URL only the *FQDN* of the hostname is valid, such as `icatserver.facility.ext` since we imported the certificate under this alias in *Glassfish* (`hostname -f`).

What’s needed

- ‘icat-setup’ Installation note icat-setup - Installation (http://icatproject.org/mvn/site/tools/icat-setup/1.1.0/installation.html)
- make the `~/java` (or `../java`) folder otherwise installation script will complain.

- icat-setup–1.1.0-distro.zip should already be downloaded
- ICAT CORE

What's to do

- unzip icat-setup–1.1.0-distro.zip in user `$HOME`,

- `./setup --verbose configure` will only create a `configured` file.

- Install

```
[glassadmin@icatserver icat-setup]$ ./setup -verbose install
```

will create sub-directory `icat-setup` in `../java` with 3 jars and an 'executable' script `icat-setup` in `~/bin`. This script starts a Java command line program.

As we said before the permission layer is empty by default and must be populated. `icat-setup` is command line utility and take some arguments to perform some tasks.

By calling

```
icat-setup https://icatserver.facility.ext:8181 db username admin password
adm_secure_password
```

one enters 'icat-setup' *interactive* mode. Here as `username` one should take an admin (listed in `rootUserMNames`) Furthermore an annoying feature of `icat-setup` is that it do not output success or failure messages. So for instance invoking a `listuser` command just after a fresh install will not echo any character (since no data in the corresponding tables is found). In interactive mode issuing an erroneous command will present a list of available commands as shown next.

```
[glassadmin@icatserver ~]$ icat-setup https://icatserver.facility.ext:8181
db username admin password adm_secure_password
No input file provided - will read from stdin
erroneous_command
Valid commands are help, adduser, addrule, addstep, listuser, listrule,
listgroup, liststep, deluser, delrule, delgroup, delstep
```

As one can see no `addgroup` keyword, the group is created implicitly with the `adduser <username> <usergroup>` command (see help).

Per default no data fill into the permissions model tables so let's create an *admin* user called `icat-admin` and as *admin* group `sysadmin`. One next must assigns permissions to the user so give him *ALL* permissions (or rules, ALL CRUD) to promulgate it to the *admin* role.

```
[glassadmin@icatserver ~]$ icat-setup https://icatserver.facility.ext:8181
db username admin password -
Please enter the value for password
No input file provided - will read from stdin
adduser icat-admin sysadmin
listuser
23 : icat-admin [sysadmin]
listgroup
22 : sysadmin [icat-admin]
listrule
addrule sysadmin ALL CRUD
listrule
25 : sysadmin Application CRUD
26 : sysadmin DataCollection CRUD
27 : sysadmin DataCollectionDatafile CRUD
28 : sysadmin DataCollectionDataset CRUD
29 : sysadmin DataCollectionParameter CRUD
30 : sysadmin Datafile CRUD
31 : sysadmin DatafileFormat CRUD
32 : sysadmin DatafileParameter CRUD
33 : sysadmin Dataset CRUD
34 : sysadmin DatasetParameter CRUD
35 : sysadmin DatasetType CRUD
36 : sysadmin Facility CRUD
37 : sysadmin FacilityCycle CRUD
38 : sysadmin Grouping CRUD
39 : sysadmin Instrument CRUD
40 : sysadmin InstrumentScientist CRUD
41 : sysadmin Investigation CRUD
42 : sysadmin InvestigationInstrument CRUD
43 : sysadmin InvestigationParameter CRUD
44 : sysadmin InvestigationType CRUD
45 : sysadmin InvestigationUser CRUD
46 : sysadmin Job CRUD
47 : sysadmin Keyword CRUD
48 : sysadmin Log CRUD
49 : sysadmin ParameterType CRUD
50 : sysadmin PermissibleStringValue CRUD
51 : sysadmin PublicStep CRUD
52 : sysadmin Publication CRUD
53 : sysadmin RelatedDatafile CRUD
54 : sysadmin Rule CRUD
55 : sysadmin Sample CRUD
56 : sysadmin SampleParameter CRUD
57 : sysadmin SampleType CRUD
58 : sysadmin Shift CRUD
59 : sysadmin Study CRUD
60 : sysadmin StudyInvestigation CRUD
61 : sysadmin User CRUD
62 : sysadmin UserGroup CRUD
```

As we just see `icat-setup` usage may be error-prone and must be handled carefully.

Component Installation : 'ICE'

Important Remarks

ICE is a Web GUI Tool used to administer *ICAT* tables limited to a small number of rows. Tables with large number of rows, such as *DATASET* or *DATAFILES* are expected[9] to be populated by 3rd party programs using the *ICAT* API.

Despite the fact that the permissions will be set accordingly after a successful login using an authentication method, the *ICE* Menu operations nevertheless are not aware of those permissions. Thus some System Error Dialog-boxes are to be expected.

It should be pointed out that it could be interesting to understand the properties defined in the `ice.properties`,since some of these properties directly influence the construction of the *ICE* **login screen** and Menus. See the *What's to do* section for more optional details.

What's needed

- *ICE* Installation Note ICE - Installation (http://icatproject.org/mvn/site/tools/ice/1.0.0/installation.html)
- ice–1.0.0-distro.zip should already be downloaded.
- ICAT Core

What's to do

- unpack ZIP and edit the `ice.properties` file and deploy the WAR with the command

```
[glassadmin@icatserver ice]$ asadmin --port 4848 deploy --contextroot ice
ice-1.0.0.war
Application deployed with name ice-1.0.0.
Command deploy executed successfully.
```

here we see that a **contextroot** must be provided and thus will define the URL for reaching the ICE WebUI as `<icat.url>/ice`, where `<icat.url>` is defined in `ice.properties`.

To `undeploy` a WAR simply issue

```
asadmin --port 4848 undeploy <war>
```

where `<war>` is the filename of the WAR file without the extension `.war` (ice–1.0.0)

Next a sample of a `ice.properties` file

```
# The server running icat
icat.url https://icatserver.facility.ext:8181/

# Space separated list of entities to be offered for editing
bean.list User UserGroup Facility InvestigationType DatasetType
DatafileFormat FacilityCycle Investigation Instrument

# Ordered list of authn methods
authn.list db simple
!ldap anon

# Ordered list of ldap credential keywords and request to hide password
!authn.ldap.list username password
!authn.ldap.password.visible false

# Ordered list of db credential keywords and request to hide password
authn.db.list username password
authn.db.password.visible false

# Ordered list of db credential keywords and request to show password
authn.simple.list username password
authn.simple.password.visible true

# Ordered list of anon credential keywords (empty)
!authn.anon.list
```

As stated in the installation's note the values of `bean.list` identify the *ICAT* Entity Beans that *ICE* will manage (will also populate the corresponding drop down list in the ICE WebUI). **If present please remove the entity Group from the list, since this is not a table found in the database**

The values of `authn.list` are used to generate a drop-down list on the *ICE* **login screen** used to select the desired authentication mechanism to identify the user. Thus the kind of credentials to supply may depend on the *authn* mechanism chosen for identification. For instance `anon` do not require any credential so no filed should be displayed.

Important : The mnemonic defined in `authn.list` must conform to the authenticator plugins installed on your system. Do not define mnemonic without having plugin installed or *ICAT* will fail to start.

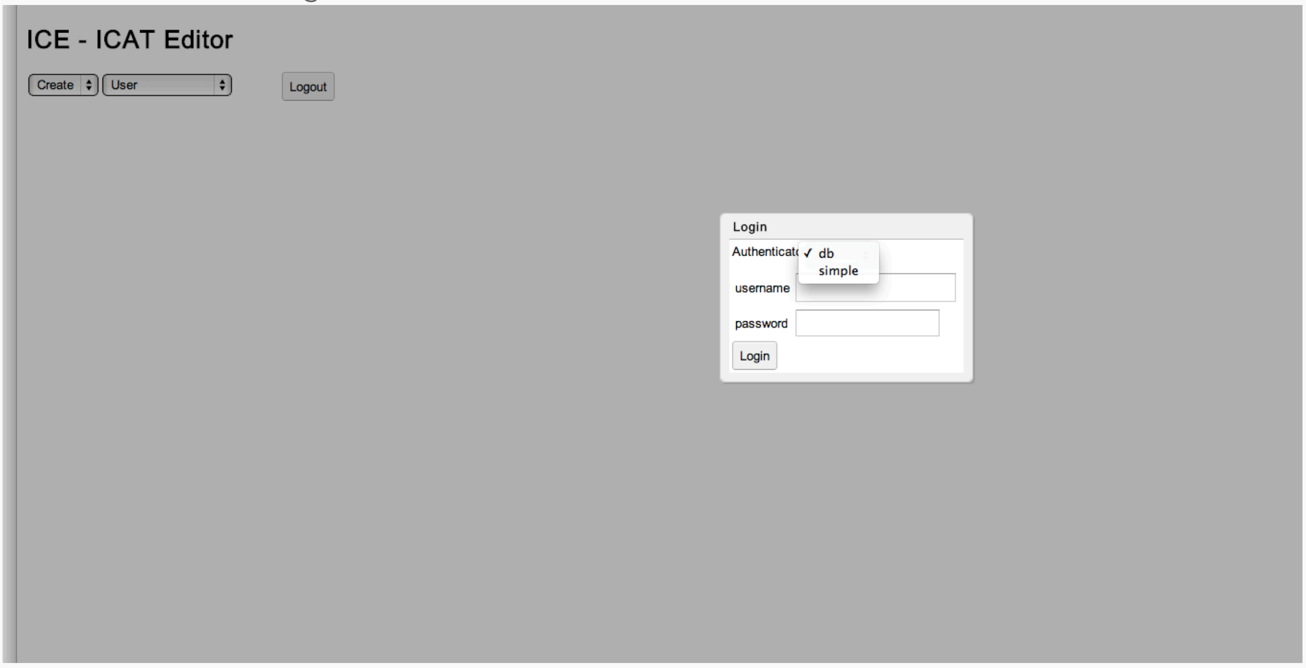
For each *authentication method* (mnemonic) defined in `authn.list`, a corresponding *'authn UI config section'* must be provided, otherwise *ICE* won't start correctly. By *'authn UI config section'* it is meant a section of two additional properties (only two, for the moment) to be supplied

- `authn.<mnemonic>.list`

- `authn.<mnemonic>.password.visible`

where `<mnemonic>` is an authentication method defined in `authn.list`.

Now go to `https://icatserver.facility.ext:8181/ice/` to login into *ICE*. Next a screen-shot of the Login Screen.



Additional Remarks about *'authn UI config section'* (Optional)

The use of the properties `authn.list` and `authn.<mnemonic>.XXX` is indeed twofold in nature. In the first place it configures the authentication method to use for accessing *ICE* and in the second place, it directly controls how the *ICE* login screen is constructed.

Indeed `authn.list` will generate UI drop-down lists, whereas `authn.<mnemonic>.list` and `authn.<mnemonic>.password.visible` will directly control the *Login Screen*. Those last two properties will generate the input text fields to gather the user credentials and also define the keywords used for passing those credentials to the *ICE* EJBs[10].

For example by defining `authn.db.list username password` *ICE* will generate two text fields on the login screen asking for a username and password, but more importantly the label of those fields will be exactly spelled as defined in `authn.db.list`.

So if one defined

```
`authn.db.list benutzer passwort`
```

then *ICE* generates two text fields named (i.e. with the label) `benutzer` and `passwort`, which could be ok **BUT** since this property also defines the associated keywords used to identify the credentials, the login process will fail. *ICE* raises an *empty password* error. And it fails because *benutzer* and *passwort* are unknown keywords to identify the `username` and `password`. So be very cautious when configuring this property.

`authn.<mnemonic>.password.visible` controls whether the password should be echoed or not.

Of course the next example would be valid

```
`authn.db.list username password1 password2 password3"
```

This snippet generates a login screen with four input text fields and also

In general each authentication method may have its own list of credentials which must be provided for a unique identification, **but** for the moment except for the `anon` method, where no credential at all is needed, only the `username` and `password` should be defined

if a deployement error happens loo at `ice.log` or `server.log`

Component Installation : TopCAT

Important Remarks

TopCAT is also Web GUI Tool (mostly written in GWT[11]) used to manage multiple ICAT services, thus allowing ICAT instances to be federated. TopCAT 1.11 only works with ICAT Core versions >= 4.3.

What's needed

- TopCAT Installation Note TopCAT - TopCAT Installation (http://icatproject.org/mvn/site/topcat/1.11.0/installation.html)
- TopCAT–1.11.0-distro.zip
- and of course a running ICAT Core installation.

What's to do

- as usual unpack ZIP and configure `topcat-setup.properties` and `topcat.properties` (`/setup --verbose configure` for assisted setup)

- configure `topcat-setup.properties`

  - uncomment and fill Oracle connection URL in *dbproperties* (SID or Service Name syntax)
  - Choose a name for the TopCAT *admin* User. Attention TopCAT **Admin password** (*adminPassword*) must contain at least one digit, a lower case character and an upper case character

- configure `topcat.properties`

  - here one can configure different URLs to official policies and and URL to the Facility Logo. The logo picture should be copied to `<glassfish_domain_dir>/applications//images

The next step is to install which will a user *adminUsername* (domain) in the ICAT Domain with group *topcatAdmin*. An entry will be found in the ICAT Domain `keyfile` (int the *<config_dir>*)

```
[glassadmin@icatserver topcat]$ ./setup --verbose install

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
property.administrative.domain.name
You are using Glassfish version 4
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 get
configs.config.server-config.security-service.activate-default-principal-to-
role-mapping
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-file-
user topcatadmin
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 --passwordfile
pw create-file-user --groups topcatAdmin topcatadmin
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
resource jdbc/TopCATDB
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 delete-jdbc-
connection-pool TopCATDB
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
connection-pool --datasourceclassname oracle.jdbc.pool.OracleDataSource --
restype javax.sql.DataSource --failconnection=true --steadypoolsize 2 --
maxpoolsize 8 --ping --property
url="jdbc\:oracle\:thin\:@//dbprd07\:1521/cusprd01.intranet.psi.ch":Implicit
CachingEnabled=true:MaxStatements=200:user=icat:password=welcome1 TopCATDB
JDBC connection pool TopCATDB created successfully.
Attempting to ping during JDBC Connection Pool Creation : TopCATDB -
Succeeded.
Command create-jdbc-connection-pool executed successfully.

execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 create-jdbc-
resource --connectionpoolid TopCATDB jdbc/TopCATDB
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy --
deploymentorder 140 TopCAT-1.11.0.war
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy --
deploymentorder 141 TopCATAdmin-1.11.0.war
```

Remark (Optional) As one can see from output the *TopCat* Admim user will be saved locally not in the *PASSWD* table. was expected since the *DB* authenticator is a plugin which be evtl. installed.

```
create-file-user --groups topcatAdmin topcatadmin
```

This options will create a user `topcatadmin`, belonging to the group `topcatAdmin`, by adding a new entry in the glassfish *keyfile* located in the `<glassfish_domain_dir>/config/` folder.

Next an excerpt of the file

```
topcatadmin;
{SSHA256}mC8mYkSI7N6)PGt71DxY&R4EGSwt7ONVp+5/rA+on2dnhf5wat+PseY/u9w==;topca
tAdmin
```

FFFFFFFFFZZZZZZZZZZZZZXXXXXXXXX @@@@@@@@@@@@

TODO encore …

```
<https://icatserver:8181/TOPCATWeb.jsp>

<https://icatserver.psi.ch:8181/TopCATAdmin/>


Idea for doing
<https://code.google.com/p/icatproject/wiki/ApacheAndGlassfish>
ApacheAndGlassfish
Apache front end for Glassfish 4
```

Short Excursions

These are only a real short excursions for recap and are not intended to be exhaustive or to replace good instructions found for instance at your local facility or on the Internet.

Configure a password-less login with OpenSSH

Using SSH for logging always ask for a password but re-entering a password for every SSH connection might be tedious with time. A way exists to speed up the process. Generating a SSH Key pair is a simple solution for entering the password only once per SSH session.

1. Think of a good password / *passphrase* since the quality of the security depends solely on the chosen passphrase.

2. first create a SSH public/private Key Pair for a dedicated `<identity`. Using the `rsa` algorithm. This will generate two key files, a public key `<identity>.pub` and a private key `<identity>` in the `.ssh` directory. If `<identity>` is not supplied (`-C <identity>` not provided)then the Key Pair generated is `id_rsa.pub` and `id_rsa`.

```
ssh-keygen -t rsa -C <idendity>
```

3. Now the public key must added to the `authorized_keys` of the user used for the installation of the ICAT server, which normally is `root`. For this purpose two possibilities

```
ssh-copy-id -i ./ssh/id_rsa.pub username@icatserver
```

and if this command is not allowed then one can use

```
`cat .ssh/id_rsa.pub | ssh username@icatserver 'cat >> .ssh/authorized_keys'
```

1. now one can use the ssh-agent for cache the passphrase (credential) into a secured memory place in order not to have each time to enter the passphrase.

```
ssh-agent -L :: to check agent status
ssh-agent $SHELL
ssh-add ~/.ssh/<indentity> :: if defaut <identity> is id_rsa
ssh username@icatserver
```

The Linux 'alternatives' System

The *alternative system* is used for managing either multiple versions of the same software or for managing multiple software offering quite the same functionality such as an editor.

Remember : The alternatives system uses symbolic links to programs offering the same kind of functionality (like several java implementations or several editors). The `alternatives` command allows to create, maintain, remove and display information about the symlinks managed by the alternatives system. Next follows some example of its usage.

```
alternatives [options] --install link <name> path priority [--slave link
name path]... [--initscript service]
alternatives [options] --remove <name> path
alternatives [options] --set <name> path
alternatives [options] --auto <name>
alternatives [options] --display <name>
alternatives [options] --config <name>
```

Where
* `<name>` is a generic name for the master link
* `link` is the master symlink, i.e. the link used by the system to start the program, such as `/usr/bin/java`.
* `path` is the real symlink to the app offering the functionality like `/usr/library/java/version/8.01/bin/java`.
* `priority` is used to determine the relative position of the *alternative* being introduced in the current priority's chain. The alternative with highest prio wins and will be started by default.

here an example with slaves

```
[root@icatserver ~]# alternatives --install /usr/bin/java java
/usr/java/latest/bin/java 1002 --slave /usr/bin/keytool keytool
/usr/java/latest/bin/keytool --slave /usr/bin/rmiregistry rmiregistry
/usr/java/latest/bin/rmiregistry
```

alternatives (symlinks) are written in `/etc/alternatives`

(Oracle) JDBC Connection Strings Handling

The ORACLE JDBC Connection Strings in *ICAT* are mostly written using the *Oracle Service Name* notation and not using the "more" common *SID* notation.

Remember the difference between the Oracle JDBC Connection Strings notations:

- `jdbc:oracle:thin:@//<host>:<port>/SERVICENAME`
- `jdbc:oracle:thin:@<host>:<port>:SID`

Furthermore since the connection strings contain symbols (in the JDBC notation itself and in the password) they must be escaped as for instance

```
icatProperties=url="jdbc\:oracle\:thin\:@//dbprd07\:1521/cusprd01.intranet.p
si.ch":ImplicitCachingEnabled=true:MaxStatements=200:user=icat:password=\"Th
e:Password?symbols\"
```

```
In the first notation the two `"'"` in the Oracle Connection String are
used for escaping the special characters.In the second notation one escapes
the colons in the string, but if the password contains symbols they must be
escaped using double quotes using both `\"` as

\"The:Password?symbols\"

If a mistake was made in the Connection URL such an error may be raised :
```

```
........

Command create-jdbc-connection-pool failed.
remote failure: Invalid property syntax, missing property value: oracle
Invalid property syntax, missing property value: oracle
Usage: create-jdbc-connection-pool
.....
```

```


# Commands Output

## CLI create-domain ICAT

[glassadmin@icatserver ~]$ asadmin create-domain icat-domain1
Enter admin user name [Enter to accept default "admin" / no password]>root
Enter the admin password [Enter to accept default of no password]>
Enter the admin password again>
Using default port 4848 for Admin.
Using default port 8080 for HTTP Instance.
Using default port 7676 for JMS.
Using default port 3700 for IIOP.
Using default port 8181 for HTTP_SSL.
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Using default port 8686 for JMX_ADMIN.
Using default port 6666 for OSGI_SHELL.
Using default port 9009 for JAVA_DEBUGGER.
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=icatserver.psi.ch,OU=GlassFish,O=Oracle Corporation,L=Santa
Clara,ST=California,C=US]
```

```
Distinguished Name of the self-signed X.509 Server Certificate is:
[CN=icatserver.psi.ch-instance,OU=GlassFish,O=Oracle Corporation,L=Santa
Clara,ST=California,C=US]
Domain icat-domain1 created.
Domain icat-domain1 admin port is 4848.
Domain icat-domain1 admin user is "root".
Command create-domain executed successfully.


## CLI start domain, enable secure admin and login

[glassadmin@icatserver ~]$ asadmin start-domain
Waiting for icat-domain1 to start ........
Successfully started the domain : icat-domain1
domain Location: /home/glassadmin/glassfish4/glassfish/domains/icat-domain1
Log File: /home/glassadmin/glassfish4/glassfish/domains/icat-
domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.

[glassadmin@icatserver ~]$ asadmin enable-secure-admin
Enter admin user name> root
Enter admin password for user "root">
You must restart all running servers for the change in secure admin to take
effect.
Command enable-secure-admin executed successfully.

[glassadmin@icatserver ~]$ asadmin login
Enter admin user name [Enter to accept default]> root
Enter admin password>
Login information relevant to admin user name [root] for host [localhost]
and admin port [4848] stored at [/home/glassadmin/.gfclient/pass]
successfully.
Make sure that this file remains protected. Information stored in this file
will be used by administration commands to manage associated domain.
Command login executed successfully.

# Generating new Glassfish certificates

TBC ..


# Debugging Help

The _ICAT_ software should normally install without big problem but when an
error is raised it is not always easy to understand what's wrong. Here we
give some examples of the problems one may encounter if one is not totally
cautious.

The problems described next were encountered during the Icat CORE
Installation.


The first example shows an error message which was appropriate but not
obvious to understand immediately

.....
```

```
execute: /home/glassadmin/glassfish4/bin/asadmin --port 4848 deploy --
deploymentorder 100 icat.ear-4.3.2.ear
remote failure: Error occurred during deployment: Exception while deploying
the app [icat.ear-4.3.2] : Exception [EclipseLink-4002] (Eclipse Persistence
Services - 2.5.0.v20130507-3faac2b):
org.eclipse.persistence.exceptions.DatabaseException
Internal Exception: java.sql.SQLException: Error in allocating a connection.
Cause: Connection could not be allocated because: Listener refused the
connection with the following error:
ORA-12514, TNS:listener does not currently know of service requested in
connect descriptor
Error Code: 0. Please see server.log for more details.

testicat copied to /home/glassadmin/bin
....
```

```
Here the Oracle _SID_ was used instead of the Oracle _Service Name_, i.e.
the JDBC URL for Oracle Service Name was used but the SID (proddb) was
given as Service Name (proddb.internal.facility.ext)


The next example is more complicated to understand for non-expert..


[glassadmin@icatserver icat]$ asadmin --port 4848 deploy --deploymentorder
100 icat.ear-4.3.2.ear
remote failure: Error occurred during deployment: Exception while loading
the app : javax.ejb.CreateException: Initialization failed for Singleton
GateKeeper. Please see server.log for more details.
Command deploy failed.


The error in this case was that the `icat.properties` was not properly
configured. Indeed the `authn.list` contained less authenticators as
enabled in the JNDI declaration section i.e. some JNDI names were not
comment out in the section.

One could avoid such an error if one carefully would have read the
Installation note. Indeed `authn.list` must only referenced plugins which
are installed and for each mnemonics defined in this list a corresponding
`authn.<mnemonic>.jndi` must be defined in the _JNDI_ section.

So this is the corrected (and valid) `icat.properties` file portion. `!` is
used to comment the rest of the line.
```

```
....
# Desired authentication plugin mnemonics
# authn.list db ldap anon simple
authn.list db simple
! ldap anon

# JNDI for each plugin
authn.db.jndi java:global/authn_db-1.1.1/DB_Authenticator
!authn.ldap.jndi java:global/authn_ldap-1.1.0/LDAP_Authenticator
!authn.anon.jndi java:global/authn_anon.ear-1.0.1/authn_anon.ejb-
```

```
1.0.1/ANON_Authenticator
authn.simple.jndi java:global/authn_simple.ear-1.0.0/authn_simple.ejb-
1.0.0/SIMPLE_Authenticator
...
```

some detailed information

here an excerpt of the `server.log`

```
.....
evelValue: 900] [[
EJB5184:A system exception occurred during an invocation on EJB
PropertyHandler, method: public java.util.Set
org.icatproject.core.manager.PropertyHandler.getRootUserNames()]]

[2014-01-24T16:45:36.602+0100] [glassfish 4.0] [WARNING] []
[javax.enterprise.system.container.ejb.com.sun.ejb.containers] [tid:
_ThreadID=1 _ThreadName=main] [timeMillis: 1390578336602] [levelValue: 900]
[[

javax.ejb.EJBException: javax.ejb.CreateException: Initialization failed for
Singleton PropertyHandler
at
com.sun.ejb.containers.AbstractSingletonContainer$SingletonContextFactory.cr
eate(AbstractSingletonContainer.java:656)
at
com.sun.ejb.containers.AbstractSingletonContainer.instantiateSingletonInstan
ce(AbstractSingletonContainer.java:396)
........


at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57
)
at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl
.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at
com.sun.enterprise.glassfish.bootstrap.GlassFishMain.main(GlassFishMain.java:
97)
at com.sun.enterprise.glassfish.bootstrap.ASMain.main(ASMain.java:54)
Caused by: javax.ejb.CreateException: Initialization failed for Singleton
PropertyHandler
at
com.sun.ejb.containers.AbstractSingletonContainer.createSingletonEJB(Abstrac
tSingletonContainer.java:483)
at
com.sun.ejb.containers.AbstractSingletonContainer.access$000(AbstractSinglet
onContainer.java:81)
at
com.sun.ejb.containers.AbstractSingletonContainer$SingletonContextFactory.cr
eate(AbstractSingletonContainer.java:654)
... 74 more
Caused by: java.lang.NullPointerException
at
```

```
org.icatproject.core.manager.PropertyHandler.init(PropertyHandler.java:100)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57
)
.......
........
Marking Tx for rollback because container for ejbName: LoggingConfigurator;
containerId: 91109222456360963 is undeployed]]

[2014-01-24T16:45:36.638+0100] [glassfish 4.0] [SEVERE] []
[javax.enterprise.system.tools.deployment.common] [tid: _ThreadID=1
_ThreadName=main] [timeMillis: 1390578336638] [levelValue: 1000] [[
Exception while invoking class org.glassfish.ejb.startup.EjbApplication
start method
javax.ejb.EJBException: javax.ejb.CreateException: Initialization failed for
Singleton GateKeeper
at
com.sun.ejb.containers.AbstractSingletonContainer$SingletonContextFactory.cr
eate(AbstractSingletonContainer.java:656)
```

`icat.log` was not really helpful and it is simple to oversee the error when reading the long `server.log` file.

Hypothetical explanation of the error: the call `org.icatproject.core.manager.PropertyHandler.getRootUserNames()` produced an exception which was **NOT catched** neither at compile-time nor at run-time since the `init()` method of the `org.icatproject.core.manager.PropertyHandler` throws an `NullPointerException`. One should perhaps improve the error handling :-)

So looking at the `icat.propeties` file because the method `getRootUserNames()` was incriminated (and the `rootUserNames` property is set there ;)) gave no hint apart from the fact that some of the authenticators were not disabled (commented out in ICAT jargon) despite that they were not listed in `authn.list` property. Thus disabling the incriminated entry in the JNDI section removed the error.

1. Red Hat Enterprise Linux ↩

2. we use the Open Source Edition of Glassfish4 ↩

3. Oracle Tech Net Instant Client downloads for Linux x86–64 (http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html) ↩

4. Oracle Glassfish Server ↩

5. Grpahical User Interface ↩

6. No scientific investigation was made only the results from our installation procedure. icat.properties defines the authentication plugins used and if not present they must be disabled (commented). Furthermore with Oracle as Back-End we had some issues with JNDI DB Pool creation. ↩

7. CRUD Create, Read, Update , Delete operations. ↩

8. Directory Information Tree ↩

9. ICE - ICE User Guide (http://icatproject.org/mvn/site/tools/ice/1.0.0/guide.html) ↩

10. Enterprise Java Beans : roughly a component controlling some application logic ↩

11. Google Web Toolkit ↩