Creating a new Aida Unix Data Provider

This wiki page describes how to create a new unix hosted AIDA Data Provider.

An AIDA "Data Provider" is a CORBA server that implements the aidaObject IDL API and registers itself with the AIDA Directory Service on an AIDA network. AIDA Data Providers typically provide high performance, robust, access to scientific data of a particular kind, like beam synchronous orbit data.

To create a new data provider on unix, we begin on an AFS unix node and use facilities in the Aida makefile system for creating the CORBA stubs and skeletons of a new Aida server.

This page goes through the steps for creating the new server called DpRdb (the Aida server for accessing a relational db), cloning the existing DpCaServer. Where in these examples you see Rdb, substitute your own server name.

First, create a development directory on AFS unix, and cvs checkout aida:

```
$ mkdir work2
$ cd work2
$ cvs co package/aida
cvs checkout: Updating package/aida
U package/aida/.classpath....
```

Source the Aida development setup files. Note the 2nd argument to aidaSetEnvDev.csh is the directory of your development instance of aida (what you created in the previous step):

```
$ source /afs/slac/g/cd/soft/dev/script/ENVS.csh
$ source /afs/slac/g/cd/soft/dev/script/aidaSetEnvDev.csh dev ${HOME}/work2/package/aida
```

cd to the dp directory

```
$ cd package/aida/edu/stanford/slac/aida/dp/
```

Create a new Data Provider (dp) directory. Then copy an existing Makefile.sun4 from one of the existing Data Provider directories, on which we'll base this new one.

```
$ mkdir dpRdb
$ cd dpRdb/
$ cp ../dpCa/Makefile.sun4 .
```

Modify the Makefile.sun4 file, replacing the PACKAGE, MODULE and INTERFACENAME values with values appropriate for your new data provider.

```
$ emacs Makefile.sun4
<edit PACKAGE, MODULE and INTERFACENAME>
```

Clone a CORBA java server entry point, from one of the other AIDA data providers. Much of the cloing can be done seding from an existing server (see sed command below), but you will have to do some edited code modifications afterward, for instance to set the SERVERNAME, change comments etc.

```
$ sed 's/Ca/Rdb/g' < ../dpCa/DpCaServer.java > DpRdbServer.java
```

Run the Makefile.sun4; this will create all client and server sides, except the _impl.java file. You will create an _impl.java file by hand later.Ignore the jidl complaint about not finding an idl file, it's not supposed to.

```
$ gmake -f Makefile.sun4
jidl: couldn't open /u/cd/xxx/work2/package/aida/idl/dpRdb.idl
sed -e 's/THIS/dpRdb/g' ....
...
```

Ok, now you're ready to start writing code! Write your implementation file. Our standard, and the makefiles assume, this will be named Dp<name>I_impl.java, eg DpRdbI_impl.java. You can ask the Aida makefile system to create a skeleton for you, or maybe you want to start by cloning an existing one:

```
Either ask Aida to make an implementation skeleton for you:
$ gmake -f Makefile.sun4 impl
Or, clone one:
$ cp ../dpCa/DpCaI_impl.java DpRdbI_impl.java
Then edit it, - change in particular the get() method
$ emacs DpRdbI_impl.java
```

Add the server to the list of Aida servers. This list is part of the Aida directory service Oracle database. See the Aida Directory Database Guide for details, off the Aida homepage. See the Aida SQL Cheatsheet, Adding a new Data Provider. It's probably a good idea to do this on AIDADEV and AIDAPROD at the same time, and check that they both get the same service ID number.

```
SQL> @add_service 'SLCBpm' 'SLC BPM orbit acquisition'
```

Find out the service ID number of the service created by the add_serices command. The ID number is computed by a database trigger. You'll use the ID when assigning AIDA names to be handled by the new service.

```
SQL> @show_services
```

Add instances and their attributes for testing. Here's a single example that may be one of thousands when it comes to deploying your server:

```
SQL>  @add_IA 103 'P2BPMLER' 'BPMS'
```

Clone a server config file, that defines on which port this server will run. The port number (defined in the line ooc.orb.oa.endpoint=iiop --port" must be unique on the host (and in fact, in case we change hosts, should at least be unique across all AIDA hosts). So, grep all the *.conf files, and pick an unused port num. By convention, the DPs are counting down from 58999. The .conf file name must be identical to the server name define by AIDA_*_NAME, since it's found automagically by the st. startup file (see below).

```
copy DpCaServer.conf to DpRdbServer.conf
<edit the file and choose a unique port number, from all the unix DPs>
```

The java command to run an AIDA server takes several arguments, to pass the values of environment variables to the server at startup, so we have st. files for each unix server. Clone one.

```
cp  st.DpCaServer st.DpRdbServer
```

Run the test server from you own account, from the host on which it will run. See the New Labour Development Cheatsheet. In particular, note that aidarun and aidaproc aliases search for Orbacus .conf files in /afs/slac/g/cd/soft/dev/confsys, so use alias aidaproclocal (see below), or not use an alias before you release the conf file.

```
alias aidaproclocal
java -server ${AIDABCSTRING} -Dooc.config=${AIDADEVROOT}/common/script/!:1.conf -
DAIDA_DATABASE_USERID=AIDA${AIDA_MODE} -DAIDA_NAMESERVER_IOR_URL=${AIDA_NAMESERVER_IOR_URL} -
DAIDA_NAMESERVER_IOR_FILE=${AIDA_NAMESERVER_IOR_FILE} -Xdebug -
Xrunjdwp:transport=dt_socket,address=!:2,server=y,suspend=n -Xms256m -Xmx256m edu.stanford.slac.aida.!:3*
```

Release

This section guides you through releasing the new data provider and its associated infrastructure.

CVS The new data provider source code

Add the new dp directory, from your area:

```
cdsoft/aida/package> cd aida/edu/stanford/slac/aida/dp
slac/aida/dp> cvs add dpRdb
```

cvs add the 3 files that make up the new server source code:

```
cvs add Makefile.sun4 DpRdbI_impl.java DpRdbServer.java
```

cvs commit them:

```
cvs commit -m "Initial version"
```

CVS the makefile additions to make the new server, and its server config file.

Working in your checked out version of common/script/:

- Add the new server (eg dpRdb) as a new target to MakefileAida.sun4 (and add it as a subtarget of the "aida" target). cvs commit MakefileAida.sun4.
- Add the .conf file (eg DpRdbServer.conf) to cvs and cvs commit it.
- Add the .conf file (eg DpRdbServer.conf) to common/script/Makefile.sun4 and commit Makefile.sun4.
- gmaketst, dev,common/script, to release the .conf file.
- If you have changed any other setup file in common/script, like aidaSetEnv.csh or aidaSetEnvDev.csh, too support runtime or build of the new server, then cvs commit it now.

Build the reference copy

Do initial checkout

```
cd $CD_SOFT/ref
cvs co package/aida/edu/stanford/slac/aida/dp/dpRdb
```

Build the new server in ref.

```
addUserRefWrite
source $AIDASCRIPT/aidaSetEnvDev.csh dev
aidamake dpRdb
removeUserRefWrite
```

We'll do "aidamake all" later (to get the new jar file, update javadoc etc), first test the examples test suite.

Create and cvs the type II startup file:

- cvs add to st.DpRdbServer to package/aida/common/script
- add st.DpRdbServer to Makefile.Host of package/aida/common/script
- add AIDA_RDB_NAME to aidamanager
- cvs commit st.DpRdServer, Makefile.Host, and aidamanager
- gmaketst, gmakedev (can't do gmakenew of a type II startup file - only systems team can do that)

Teach procmanager and warmst about the new server

In common/setup/ edit setEnv.csh - add AIDA_RDB_NAME, AIDA_RDB_HOST_DEV etc section (see other aida servers for * reference), then cvs and release it:

```
cvs commit setEnv.csh
gmaketst common/setup
gmakedev common/setup
```

In common/tool/ add the new server to procmanager and warmst, then cvs commit and release them. You have to release all teh way to new, because cddev does not have the "dev" directory in its path. So we can't test warmst until it's all the way to new.

- To procmanager script, add line for AIDA_RDB_NAME
- To warmst script, add line for new server
- cvs commit procmanager and warmst
- gmaketst common/tool, gmakedev common/tool, gmakenew common/tool.

Test starting the new server

In a new login on a development host (so as to pick up the additions to the environment released above):

- source /afs/slac/g/cd/soft/dev/script/ENVS.csh
- source !:1:h/aidaSetEnv.csh dev
- kinit (to get a token for the ssh done in the aidamanager step)
- aidamanager $AIDA_RDB_NAME start dev
  You may like to have the cmlogviewer running while you do this.

Run the test suite, release it, and clean up

- Run the test suite you created for the server (as yet unreleaed) on the released server.
- aidatest dpRdbTest 1 2 3 etc
  Then release
- java/
  - cvs add
  - cvs commit
- matlab/
  - test the array extraction in matlab.
  - cvs add
  - cvs commit
    Compile the test suite in the released area:

```
> addUserRefWrite
> source $AIDASCRIPT/aidaSetEnvDev.csh dev
> aidamake testsuite
> removeUserRefWrite
```

Finally, aidamake all, to make the javadoc, zip file for Eclipse etc.
Test from the ref test examples.

Email Jingchen, ask him nicely to

- gmakenew package/aida/common/script, to new the st file
- create the "type I" st. file

Ask Bob nicely to:

- add DpRdb to server check
- add DpRdb to UWD