Aida Directory Service Management Tools

Contents: SQL Tools, Administration

SQL Tools

A number of SQL scripts are provided for Aida directory service management. These are in the CD_SOFT SQl script release directory "/afs/slac/g/cd/soft /dev/ora/". That directory is in ORACLE_PATH, which is set by sourcing /afs/slac/g/cd/soft/dev/script/oracle_env.csh.

The Aida Directory Service database schema is given here.

Setup

To set yourself up to execute these tools, log into a SLAC Public AFS Solaris machine (Tersk, Flora etc) and then, first, set your environment:

```
[tersk02]:u/cd/greg> source /afs/slac/g/cd/soft/dev/script/ENVS.csh
[tersk02]:u/cd/greg> source /afs/slac/g/cd/soft/dev/script/aidaSetEnv.csh [DEV|PROD|LCLSDEV|LCLS]
```

Make sure you have TWO_TASK set appropriately:

[tersk02]:u/cd/greg> printenv TWO_TASK

DEV=AIDADEV@SLACPROD
PROD=AIDAPROD@SLACPROD
LCLSDEV=AIDA@MCCQA
LCLS=AIDA@MCCO

(See these defined in /afs/slac/g/cd/soft/ref/package/aida/common/script/aidaSetEnv.csh)

WHEN FIRST ADDING A NAME, USE "DEV". That will put the new name in AIDADEV. So you will then check that the AIDADEV network can get a value for that name. After you've verified it all works in AIDADEV, repeat for AIDAPROD, test again, and finally put the name in MCCQA. After the Synch to MCCO, AIDALCLS will be able to perform the query for that name.

Then enter a sql interpreter like sqlplus. You give the Aida database on which you will be performing operations. On tersk this is:

```
sqlplus AIDADEV/<password> or
sqlplus AIDAPROD/<password>  // same password for AIDADEV and AIDAPROD
```

You can also use emacs' sql-oracle mode. You'll need to ask an Oracle administrator for the password.

Adding a Name

First, find out the AIDA Server id number for which you'll be adding the name. To do that eithe rjust look in AIDA_SERVICES table yourself, or use the script show_services.

Second, add a row to AIDA_NAMES and AIDA_DIRECTORY to implement the new entity, per the schema defined in the Aida Schema. Effectively, do what is defined in /afs/slac/g/cd/soft/ref/package/aida/common/script/add_IA.sql or add_IAT.sql.

Management SQL scripts

This subsection describes the use of some of the many SQL scripts that you can find in /afs/slac/g/cd/soft/package/aida/common/script/.

Show Aida Services

To show names and ID numbers of all Aida services in the Aida directory service:

```
@show_services
```

This can be executed from the command line with "show_services".

To Add a Name to Aida.

To add a name to Aida, run add_IA.sql, giving the number of the Aida service to which you want to add the name, plus the instance and attribute of the name:

```
@add_IA 63 'BPMS:PR02:8032' 'TWISS'
```

You can find all the existing Aida service names with show_services.sql. Add the name first to AIDADEV, then test; if successful, add the name to AIDAPROD.

add_IA can be used from the command line to execute add_IA.sql.

To Add a Name with a Transform to Aida

A "transform" is a pattern that is sent to the data provider in place of the given name. It's used for instance to provide the SQL query that the RDB data provider should run to get the data for a name. To add a name and transform, use add_IAT.sql. The following example tells Aida to pass a sql select SELECT statement to the Database data provide (number 5 - see show_services), when asked for BPMS:IN20//BSA.X1H.NAMES.

```
@add_IAT 5 "BPMS:IN20" "BSA.X1H.NAMES" "select unique instance from aida_names where instance like ''BPMS:
IN20:%:X1H''"
```

Note the use of quotes - you must quote the transform so it's a single argument to add_IAT.sql, and if the transform itself includes quotes as this one does, then they must be double-quoted in sqlplus.
Add the name first to AIDADEV, then test; if successful, add the name to AIDAPROD.

Add a transform that includes a replacement pattern

The AIDA Name Server supports pattern replacement, to allow a transform to be written using the literal strings &instance and &attribute to stand for the instance and attribute of the aida name of the transform. This allows the same transform to be used for many instances and attributes. Note that, for efficiency of runtime processing, transforms that require pattern replacement must have '/' as their first character, so the AIDA directory service knows only to do the expensive pattern replacement after examining the first character. The second thing to note when adding transforms to the AIDA directory service with SQLPLus (as add_IA does), occurrances of & must be escaped. Putting these two considerations together here's an example: the following name and transform will allow AIDA to acquire the SUML_M (the sum of the effective lengths up to the device) associated with XCOR:IN20:121. And the same transform would work for all devices.

```
add_IA 5 XCOR:IN20:121 suml_m "/select suml_m from
lcls_infrastructure.v_lcls_elements_report where epics_device_name = ''\&instance''"
```

This will result in the following query text sent to service 5 (the Oracle data provider):

```
select suml_m from lcls_infrastructure.v_lcls_elements_report where epics_device_name = 'XCOR:IN20:121'
```

Verify presence of a name

To verify existence of an aida name, and to get its AIDA_NAMES table ID:

```
@show_IA 'BPMS:IA20:235' 'twiss'
```

show_IA can be executed from the command line. To use wildcards, or get the transform etc, use the APEX interface.

Show everything about an AIDA name

Use show_all. This example is more sophisticated, showing how to get information about all the twiss names of all the instances in a file, and recording results to a file:

```
cat ~/checkDevices27Oct2008.txt_Aida_NO | xargs -I {} -t show_all {} twiss >& show_output.txt
or, using awk to select a field from a file
awk '{print $1}' < checkDevices27Oct2008.txt_lexid_DPonDEV | xargs -I {} -t show_all {} twiss
```

To Remove a Name from Aida.

To remove an name from Aida, run remove_IA.sql, giving the name's instance and attribute:

```
SQL> @remove_IA 'LCLS' 'BSA.rootnames.byZ'
```

Remove the name first from AIDADEV, then test, then it from AIDAPROD too.
remove_IA can be executed from the command line.

To Change the Transform of a Name

To change the transform Aida sends a data provider for a given instance-attribute name, use update_IAT.sql, giving the instance, attribute, and new transform. The following example changed the transform associated with the Aida name LCLS//BSA.elements.byZ:

```
@update_IAT 'LCLS' 'BSA.elements.byZ' 'SELECT ROOT_NAME, SOURCE, ELEMENT, BEAMLINE, AREA, LINACZ_M FROM
LCLS_INFRASTRUCTURE.V_LCLS_BSA ORDER BY LINACZ_M'
```

Change the transform first in AIDADEV, then test; if successful, change the transform in AIDAPROD.
replace_IA can be used at the command line.

To Add a Transform to an Existing Name

To make Aida send a transform to the data provider for a given instance-attribute name, use update_IAT.sql, just as you would have had the name had a transform associated with it before.
Add the transform first in AIDADEV, then test; if successful, add the transform in AIDAPROD.

Execute aida_xalServ_names_update by hand.

`aida_xalServ_names_update` looks at the XAL model devices in MACHINE_MODEL schema on the instance on which it's executed, and updates the given AIDA schema on that instance, with any changes it finds.

So, it's run automatically by ModelManager after a model has been uploaded that contains new devices or devices that have changed name.

If it has to be run by hand
Log into MACHINE_MODEL on the instance whose AIDA schema you want to update, eg AIDADEV, AIDAPROD, MCCQA:

```
SQL> SET SERVEROUTPUT ON
SQL> set timing on
SQL> SPOOL RUN_AIDA_XALSERV_NAMES_UPDATE_MCCO.LOG
```

Execute it, with last arg 'P' for pending.

```
SQL> EXEC ONLINE_MODEL_PKG.AIDA_XALSERV_NAMES_UPDATE('AIDA', '202', '202', 'P');
SQL> SPOOL OFF;
```

Then check the results. If the AIDA schema changes check out, then commit the change.

```
SQL> select n.instance, n.attribute, s.name, n.transform from
aida.aida_names n, aida.aida_directory d, aida.aida_services s where
n.instance like '&1' and n.attribute like '&2' and n.id = d.name_id and d.service_id = s.id;
SQL> commit;
```

Check the results actually by getting model.

Execute aida_symbols_names_update by hand.

`aida_symbols_names_update` updates attributes like element.effective_length and element.S, used by Henrik's model applications.

At the time of writing, it is still only executed by hand following a presumed update in LCLS_INFRASTRUCTURE. That has to be fixed.

If it has to be run by hand
Log into MACHINE_MODEL on the instance whose AIDA schema you want to update, eg AIDADEV, AIDAPROD, MCCQA:

```
SET SERVEROUTPUT ON
set timing on
SPOOL RUN_AIDA_SYMBOLS_NAMES_UPDATE_MCCO.LOG
```

Execute it, with last arg 'P' for pending. **Replace the schema name** with the appropriate one for the instance on which you're running.

```
EXEC ONLINE_MODEL_PKG.AIDA_SYMBOLS_NAMES_UPDATE('AIDA', '5', '5', 'P');
SQL> SPOOL OFF;
```

Then check the results. If the AIDA schema changes check out, then commit the change. Eg, on MCCO the aida schema is called AIDA, so:

```
define I='YCOR:LI30:900'
define A='element.effective_length'
select n.instance, n.attribute, s.name, n.transform from
AIDA.aida_names n, AIDA.aida_directory d, AIDA.aida_services s where
n.instance like '&I' and n.attribute like '&A' and n.id = d.name_id and d.service_id = s.id;
SQL> commit;
```

Check the results by aidaget. Eg: `YCOR:LI30:900//element.effective_length`. If you have a file of the names for which to get data, you can check them all with something like;

```
cat uniq_mag_names.txt | xargs -t -I{} aidaget {}//element.S
```

Synchronize MCCQA to MCCO

On a Solaris host, after AIDA development environment setup, execute:

```
aida_validate_and_sync_instances.sh 'what, eg SLC Names'
```

or, in SQL:

```
SQL> @aida_qatoo_sync.sql 'what, eg SLC names'
```

Administration

This section presents documentation useful to the Database administrators and developers responsible for the maintenance of the Aida directory service. This information is at a lower level than the SQL Tools presented above.

The Aida Directory Service database schema AIDA_Dir_Service_Db_Schema.pdf.

The Schema DDL of AIDADEV (identical to AIDAPROD) contains a listing of all table definitions, triggers and Stored Procedure listings used by AIDA.

aida_schema_ddl.sql