Easy TeraGrid Accounting: Amie / Gold Integration

authors: Steven Brandt, Daniel S. Katz, Michael Shapiro

1 Abstract

Adding a resource to the TeraGrid can be a difficult task for an institution, particularly if the institution is also new to the TeraGrid. Two of the more challenging aspects are developing a local accounting system (if the site does not already have one) and then integrating that local accounting system with AMIE (Account Management Information Exchange). In order to make this task much more simple, LONI/LSU/CCT has developed a bridge between AMIE and the popular free allocation system named GOLD that has been developed by Cluster Resources. This bridge software, AmieGold, automates the testing and production phases of integration, processing of all TeraGrid account and allocation requests.

2 Introduction

The TeraGrid is an open scientific discovery infrastructure combining leadership class resources at eleven partner sites to create an integrated, persistent computational resource.

Using high-performance network connections, the TeraGrid integrates high-performance computers, data resources and tools, and high-end experimental facilities around the United States. Currently, TeraGrid resources include more than 750 teraflops of computing capability and more than 30 petabytes of online and archival data storage, with rapid access and retrieval over high-performance networks. Researchers can also access more than 100 discipline-specific databases. With this combination of resources, the TeraGrid is the world's largest, most comprehensive distributed cyberinfrastructure for open scientific research.

The TeraGrid is coordinated through the Grid Infrastructure Group (GIG) at the University of Chicago, working in partnership with the Resource Provider sites: Indiana University, the Louisiana Optical Network Initiative, National Center for Atmospheric Research, National Center for Supercomputing Applications, the National Institute for Computational Sciences, Oak Ridge National Laboratory, the Pittsburgh Supercomputing Center, Purdue University, San Diego Supercomputer Center, Texas Advanced Computing Center, and University of Chicago/Argonne National Laboratory.

TeraGrid users apply for resource allocations, including computer time, storage, and advanced user support. These applications are reviewed by a peer-review committee, which then leads to some applicants receiving grants of resources. In the case of computer time, the amount of time awarded on each TeraGrid machine is communicated through AMIE.

AMIE was created jointly by Boston University's Scientific Computing and Visualization group and the National Center for Supercomputing Applications. It provides a method for exchanging information between distributed resources to enable grid-based allocation management. It is a component that fits between local allocation systems, allowing them to communicate. AMIE is constructed from open source and highly portable languages such as Perl, XML, and SQL.

When a new resource is added to the TeraGrid, it must integrate to AMIE [**?**]. The TeraGrid does not seek to impose an accounting system on individual sites, understanding that they may already have such a system in place that does the local job well. It does, however, require that the local accounting system be able to speak to AMIE.

AmieGold was designed to address the needs of a resource provider that did not already have such a system for their resource. It is intended to be an an off-the-shelf component providing local allocation management and TeraGrid connectivity. AmieGold is a minimal piece of software that links two standard packages together: AMEIDB.pm, a perl API, and Gold [**?**], a popular allocations system provided by Cluster Resources.

The AMIEDB.pm perl module provides an interface to the AMIE reference implementation database. It allows the programmer to interact with the reference implementation database without having to write SQL. It provides methods to retrieve AMIE packets and their data, as well as methods to create AMIE transactions and packets.

The Gold Allocation Manager was developed by Pacific Northwest National Laboratory (PNNL). Among its many features, it supports rationing of resources by project, keeping track of jobs and providing access to historical usage records. The gold package is highly customizable, enabling local sites to keep track of types of special data.

Gold is actively supported and is available from Cluster Resources. While the software itself is free, support can be purchased. Gold can directly integrate with Moab [**?**], Cluster Resources flagship product, but AmieGold provides its own mechanism, preferring to interface with the resource manager (Torque [**?**] or Loadleveler [**?**].

3 The Test Harness

The AMIE system comes with a test harness, which is basically a closed test environment that will allow the system to be tested without interfering with any ongoing operations of the computing system on which AMIE is installed. All new TeraGrid systems must be able to process the packets in the test harness before they will be connected to the TeraGrid central database (TGCDB). This paper walks the reader through the installation of AmieGold, and the running of AmieGold through the test harness. When this process is complete, the following users will have been added to the system: postgres, gold, amie, tgcdb. It is possible to use a different database instead of postgres (such as mysql) but one still needs to install postgres for use by the test user named tgcdb.

3.1 fedora

The following guidelines apply to Fedora and its "yum" distribution system, but it should be relatively straightforward to map the requirements below to other opterating systems.

Basic packages:

```
yum -y install perl-DBD-Pg \
    perl-XML-LibXML \
    perl-XML-XPath \
    perl-XML-Writer \
    perl-Log-Log4perl \
    perl-Error \
    perl-Data-Properties \
    perl-Crypt-CBC \
    perl-Digest-SHA1 \
    perl-Digest-HMAC perl-suidperl
```

These packages will help with a manual install of postgres:

```
yum -y install perl-ExtUtils-Embed \
    readline-devel zlib-devel gcc \
    bison flex
```

These are just generally useful:

```
yum -y install wget svn
```

3.2 POSTGRES

Ideally one should use postgres 8.1 as the TGCDB code has not been officially tested under later versions. In addition, the plperl and plpgsql language extensions have to be installed.

To obtain the plperl language, one needs to do one of the following:

- For a source build: Include the "–with-perl" option on the configure line. Before you configure, make sure "perl -MExtUtils::Embed -e ccopts -e ldopts" works.
- For a binary install: Install the postgres-pl package.

One should perform the following commands as the postgres user if the postgres database will be used exclusively. If mysql (or something else) is substituted for the amie and/or gold users' databases then these scripts and instructions will not work unmodified. Changes should be straightforward, however.

- gold - These commands will prepare the postgres database for the Gold installation.

  ```
  createuser gold # answer no to questions
  createdb gold --owner gold
  ```

- amie - This user will provide the connection to the TeraGrid Central Database. For the purposes of running the test harness, the local user named "tgcdb" will pretend to be the TeraGrid Central Database.

  ```
  createuser amie # answer no to questions
  createdb amie --owner amie
  ```

- tgcdb - The TeraGrid Central Database test harness user.

  ```
  createuser tgcdb # answer no to questions
  createdb tgcdb --owner tgcdb
  createlang plperl -d tgcdb
  createlang plpgsql -d tgcdb
  ```

3.3 AMIE

The amie user will require the ability to ssh into the tgcdb user, and vice versa. Amie will also need to be able to ssh to itself. The user installing AmieGold needs to issue the following commands as root:

```
cat ~tgcdb/.ssh/id_rsa.pub >> ~amie/.ssh/authorized_keys
cat ~amie/.ssh/id_rsa.pub >> ~tgcdb/.ssh/authorized_keys
cat ~amie/.ssh/id_rsa.pub >> ~amie/.ssh/authorized_keys
```

Next, as amie, the user should run these commands:

```
mkdir repos
cd repos
wget http://software.teragrid.org/tgcdb/amiegold.tar
tar xvf amiegold.tar
# or you could do this:
# svn checkout https://svn.cct.lsu.edu/repos/scss/amiegold
```

The user should the edit scripts/conf/amie/amie.config and scripts/conf/tgcdb/amie.config, changing the admin.email parameter to something appropriate for the installation. If the AMIE system encounters problems during the execution of the test harness or later on during production it will generate emails and send them to this account. Please check email after running your test harness. A lack of email from amie/tgcdb will indicate success.

Next you need to edit the amiegoldcfg.pl script. You need to tell the script the proper way to connect to your databases, the name of your machine, and the name of your site.

The amiegoldcfg.pl perl script should now be run as follows:

```
cd ~/repos/amiegold/trunk/amieimplement
perl scripts/amiegoldcfg.pl
```

Running this command will generate most of the configuration scripts and files needed to complete the testing.

The amiecfg.sh command should now be executed as the amie user. It carries out the following configuration tasks:

- Create necessary directories: The amie configuration needs the directories amie-local/xfer, amie-local/logs, and amie-local/conf. The conf directory will be automatically populated. It will contain a db.config, telling AMIE how to connect to the database, and amie.config, which supplies information on how to connect to the TeraGrid. The initial configuration will point to our test version of the TGCDB. The amie-local/xfer directory will contain a file for each AMIE packet sent to and received from the TGCDB in XML format. If a problem were to arise at some point, these files could be inspected to facilitate debugging.
- Downloading the AMIE code: The AMIE code comes in three parts and is available from software.teragrid.org. The amiecfg.sh will retrieve them via wget.
- Configuring the database connection: The script will create ∼amie/repos/tgam/data/dsn/tg-dev.dsn, enabling connection to the amie database.

Please source your .bashrc file after running the script to pick up changes to your path.

3.4 GOLD

The Gold package can be obtained from Cluster Resources through the link in reference [**?**].

After successfully installing Gold, it is necessary to run the gold extension script.

```
sh goldext.sh
```

This file will create the amie user in gold, assign it administrative privileges, and create additional fields in the gold job records, etc. that are required by the TeraGrid.

The gold user should have the following added to its .bashrc:

```
export GOLD_HOME=/usr/local/gold
export PATH=$PATH:$GOLD_HOME/bin:$GOLD_HOME/sbin
```

Subsequent to successfully installing gold, you will need to make sure that gold commands are in the path (i.e. PATH environment variable) and can be run as the gold, amie, and tgcdb users. You can run the "glsuser" command as an example.

3.5 TGCDB

The tgcdbcfg.sh command should be executed as the tgcdb user. Please note that the tgcdb user needs permission to read from the ~amie/repos directories.

Note that tgcdbcfg.sh will modify the tgcdb user's bashrc file. Please source the .bashrc file after running the script to pick up the changes to the PATH.

The person installing AmieGold at another site should work with the TeraGrid to finalize local settings before going live.

3.6 RUNNING THE TEST HARNESS

At this point it should be possible to run the test harness. To do this, the person installing AmieGold should execute the program test-suite.sh as amie in the ~amie/repos/amiegold/trunk/amieimplement directory. This will load a variety of packets and test out the installation in a number of ways.

The test-suite.sh program performs several functions:

- It begins by clears out test users and test accounts out of the Gold database, deleting all files in the ~/amie-local/xfer directories for both the amie and the tgcdb users, dropping all data in the local AMIE database, and deleting all functions and tables from the tgcdb user database. It then and re-creates the tgcdb database from scratch. If this is the first time test-suite.sh is being run, then these tables and functions will be created and initialized.

- Test packets of type request_project_create packet and request_account_create are loaded into the database. two test users.

- The program cycle.sh (which calls the "amie" program as both the amie and the tgcdb user, runs amiegold.sh, then prints out the state of the AMIE and TGCDB databases) is called, and amiegold.sh creates two test users in an inactive state.

- Explicit gold commands are now run to activate these users.

- The program cycle.sh is run until the project and account creation are complete.

- Explicit gold commands are now run to charge for a job.

- The program cycle.sh is called to send the usage packet to the TGCDB database.

- Two run packets are processed, a 'replace' and a 'delete' packet. The former augments the DnList. The latter replaces it.

- The program cycle.sh is called until the packets are processed. The query "goldsh DnList Query" is called at various stages to show the changes to the DnList.

- Test packets of type request_project_inactivate packet and request_account_inactivate are loaded into the database.

- The program cycle.sh is called until the packets are resolved. At the end of the run, the DnList has only three entries and the test project has no members.

NOTE: The test-suite should never be run on a production system, as it cleans out xfer files and the database itself.

After a successful completion of test-suite.sh, execution the ddump.sh command (a tool for displaying a snapshot of the packets in the database) should produce something similar to what is shown in Figure 1.

The salient point to notice is that all the important types of packets have completed at the end of the test (some of the inform_transaction_complete packets will be marked "in-progress," however, and that is acceptable).

This test verifies the following operations:

- request_project_create,
  notify_project_create,
  data_project_create,
  inform_transaction_complete.
  This set of packets creates a project and the login for the PI.

- request_account_create,
  notify_account_create,
  data_account_create,
  inform_transaction_complete.
  This set of packets creates a login.

- notify_project_usage,
  inform_transaction_complete.
  This packet set is used to send Job records back to TeraGrid. It can process both charges and refunds.

- request_user_modify,
  inform_transaction_complete.
  This packet set is used by TeraGrid to update account information and the Globus DN list. There are two different types of RUM packets, and both are tested here.

- request_account_inactivate,
  notify_account_inactivate,
  inform_transaction_complete.
  This packet set will remove an individual user from a Gold project.

- request_project_inactivate,
  notify_project_inactivate,
  inform_transaction_complete.
  This packet set will remove all users from the corresponding Gold project, effectively invalidating it.

4 USING THE SYSTEM IN PRODUCTION

4.1 SETTING UP A PRODUCTION SYSTEM

When a site has finished implementation and local testing, the final steps are:

- Get the site and resource(s) defined in the TGCDB.

- Setup an AMIE connection with the TGCDB.

- Process a single RPC/RAC/etc transaction sent by the TGCDB, as well as send a single NPU. It is possible to configure AmieGold.pl to process a single packet at a time. Simply modify the $iters variable to be one.

  ```
  my $iters = 1;
  ```

  To test sending back an NPU, create a job by hand using Gold's gcharge command. A sample invocation may be found inside the test-suite.sh script.

These actions will require coordination with the administrators of the TGCDB. These details of these steps are beyond the scope of this paper.

4.1.1 Cron

Next it will be necessary to get both the amie and amiegold.sh scripts into cron, preferably in a single script. An example of such a script is:

```
#!/bin/bash
amiegold.sh
amie
```

The amie command handles the pushing and pulling of AMIE packets to and from TeraGrid central. The amiegold.sh script will exchange data between the amie database and the Gold database.

4.1.2 The grid-map file

AmieGrid does not create or set passwords, it instead requires grid certificates for management of user logins. This simplifies administration considerably – passwords don't need to be generated and sent by mail, nor re-distributed by the TeraGrid.

This means, however, that the grid-map file needs to be kept current with data from the TeraGrid as well as any sources local to each organization. The script "mkgridmapfile.py," written by Michael Lambert at LSU, is provided to minimize the overhead of maintaining the grid-map file. This script can query multiple grid-map file sources, merging them into a single file. These sources can take the form of local files, data exposed via HTTP or system commands.

This script should be run automatically and frequently as TG grid-map file data has been known to change rapidly, up to several times per day.

The basic data used to generate the grid-map file can be obtained by running the Gold command below:

```
goldsh DnList Query --raw
```

The output will contain a list of user names and Globus DNs delimited with the pipe symbol (except the first line which supplies column titles).

```
User|Dn
ngnedin|/C=US/O=Some other Fake Org/CN=Nickolay Y. Gnedin
ngnedin|/C=US/O=Fake DN Organization/CN=Nickolay Y. Gnedin
dnagai|/C=US/O=NPACI/OU=SDSC/CN=Zoran Mikic/UID=u1462
```

4.2 Running in Production

4.2.1 Logins

What login name does a TeraGrid user receive? The AMIE packets supply a list of names which may be used by AmieGold to generate an account. AmieGold will use them if:

- the name does not conflict with any existing user name in the Gold database.
- the name is 8 characters long or less.

If neither condition is met, or if no login name is suggested inside the AMIE request, then AmieGold will construct a login from the user's first and last name, and possibly a number.

4.2.2 Approving Users

The TeraGrid requires that user logins not be duplicated. To support this requirement, AmieGold creates Gold users in an inactive state. AmieGold will not create projects for inactive users, so the processing of a request_project_create packet will be suspended until its PI is activated.

One can list inactive users by issuing the following command:

```
glsuser -I --show Name,TgPersonId
```

The output should look something like this:

```
Name    TgPersonId
------- ----------
dnagai  2
ngnedin 1
johndoe 347
...
```

If it is certain that a given Gold user does not yet have a login, that user can be activated using the following simple Gold command:

```
glsuser -A johndoe
```

If the user already has a login, it will be necessary to supply that user with the correct TgPersonId. For the sake of this example, let's assume that there is already a user named jdoe on the system. TeraGrid now sends a packet for johndoe.

Are jdoe and johndoe the same person? It is possible to use gold to obtain more detailed information about them.

```
glsuser jdoe --show Name,FirstName,LastName,EmailAddress
```

This command will produce output similar to the following:

```
Name   FirstName LastName EmailAddress
------ --------- -------- ---------------------
jdoe   John Q.   Doe      jdoe@somewhere.edu
```

The full range of fields that are currently supported for querying are as follows: Name, Description, CommonName, PhoneNumber, EmailAddress, DefaultProject, OfficeAddress, BzPhoneExt, Citizenship, FirstName, StreetAddress2, City, State, Title, Dept, LastName, Zip, MiddleName, Org, Position, HomePhone, Country, StreetAddress, TgPersonId, and Fax.

If it is determined that jdoe and johndoe are the same person, then the original user (jdoe) must be identified as being a TeraGrid user and the new user (johndoe) must be deleted. This is accomplished by adding the appropriate TgPersonId (see the listing above for the value in this example) and deleting the redundant login.

```
gchuser jdoe -X TgPersonId=347
grmuser johndoe
```

Once there is an active user with the appropriate TgPersonId AmieGold will continue the processing of packets.

4.2.3 Approving Batches of Users

If the system is exclusively for TeraGrid users, then one can simply activate all Gold users.

After users are activated, they can be added to the local ldap, or /etc/passwd file, or whatever local mechanism is being used to manage logins.

A sample shell script that could be used in this case looks something like this. the following:

```
#!/bin/bash
for u in $(glsuser -I --raw --show Name | grep -v Name)
do
   useradd $u
   sudo -u gold gchuser -A $u
done
```

If there are local users in addition to TeraGrid users, and logins are administered using LDAP, then one may wish to consider using CySolidus [**?**] to facilitate finding matches between existing users on the system and handling these cases appropriately.

Documentation and installation instructions for CySolidus are currently being compiled and it is hoped that they will be available soon.

4.2.4 Querying Jobs

The glsjob command is the primary tool in gold for querying jobs.

```
glsjob -s '2008-04-01' -e '2008-04-04' -u jdoe
```

The above command will list all jobs starting on or after April 1 and ending before April 4 that were run by user jdoe. If you run glsjob without any arguments it will list all the jobs in the database, and this may take a very long time if your database is large.

4.2.5 Refunding Jobs

If a job fails due to a system error, the local administrators might want to refund the time used in the failed job. This can be done through Gold using the standard refund command.

```
grefund -J 12345.host -d ``TeraGrid ticket #6789''
```

The next time the AmieGold commands run, the presence of a new refund record in the Gold database will be detected and a notify_project_usage packet will be generated and sent to the TeraGrid. The description field of the refund will be copied into the TeraGrid packet's comment field. It is recommended that this comment always contain an appropriate TeraGrid ticket number.

4.3 Reporting

Although the Gold software can report on balances, remaining time, and jobs that have been executed against an allocation, the information available in the AmieGold system will be incomplete for roaming or multi-machine accounts. TeraGrid does not push that kind of usage information out to individual sites.

For this reason users should not rely on gbalance to examine the state of their TeraGrid projects but should instead use the tg-usage command that is part of the TeraGrid software stack (CTSS) [**?**]

4.4 Coexisting with Local Allocations

All TeraGrid allocations will begin with the prefix "TG-". If local allocations systems use a different prefix, local allocation can co-exist with TeraGrid allocation on the same system, and collisions can be prevented.

5 Demonstration

Discounting the time to install Postgres and Gold, it is possible to install AmieGold and get the test harness running inside of ten minutes, and that is the demo that is planned for TeraGrid 08. Getting the test harness running with AmieGold should be one of the least difficult tasks of joining the TeraGrid.

6 Future Work

Although Gold has hooks in Moab for calling the Gold command gcharge, it may not be the appropriate choice for a system running AmieGold for the following reasons:

- Custom fields: The Moab hook does not include all the information needed by TeraGrid. For example the number of nodes used during the computation. Nodes is a required field in the notify_project_usage packet.
- Outages: If the Gold database goes down then queued jobs may be blocked or delayed. Even if the outage does not cause this effect, charges that occur during the outage will be missing from the Gold database. An implementation that reads log files, and which runs outside of the scheduler, provides an answer to both of these concerns. Because it runs outside of the scheduler, it cannot block its operations. Because it obtains its data from log files, it can easily replay job completions that occurred while the system was down.
- Enforcement: Gold can enforce allocations in a number of ways, but the designers of AmieGold believe that the best policy is to block jobs at submit time or not at all. Both Torque and Loadleveler provide a filter mechanism which can be used to perform this task. The filter checks against allocation data stored in a flat file that is periodically updated with allocation and user data, ensuring that problems affecting the Gold database or server process will not cause job submissions to be blocked or delayed.

AmieGold comes with a handful of scripts that implement the alternative charging and enforcement scheme described above, and it is planned to complete documentation for them soon. For the moment they are available in the scripts/charging directory of the AmieGold distribution, but at a later time they may move to their own repository.

These issues will be addressed in a future release.

7 Conclusions

AmieGold is currently being used in production on LONI's Queen Bee, and has been installed and tested on NICS's Kraken, both TeraGrid systems. In the production installation at LONI, over 5,000 packets have been received as of late March 2008, and an equivalent number packets have been generated and transmitted to AMIE. The AmieGold package was installed, and the test harness successful executed, by Victor Hazlewood at Oak Ridge National Laboratory in preparation for joining Kraken to the TeraGrid. These examples show that the AmieGold software is working correctly and can be installed by a non-developer. LONI/LSU/CCT will continue to maintain this software to match any changes that occur in AMIE, and will eagerly seek additional AmieGold users.

References

[1] AMIE. http://scv.bu.edu/AMIE/.

[2] Gold. http://www.clusterresources.com/pages/products/gold-allocation-manager.php.

[3] LoadLeveler. http://www-306.ibm.com/software/tivoli/products/scheduler-loadleveler/.

[4] MOAB. http://www.clusterresources.com/pages/products/moab-utilityhosting-suite.php.

[5] CySolidus repository. https://svn.cct.lsu.edu/trac/scss-weblication/.

[6] Common TeraGrid Software Stack. http://www.teragrid.org/userinfo/software/ctss.php.

[7] Torque. http://www.clusterresources.com/pages/products/torque-resource-manager.php.

```
amie
 packet_rec_id | trans_rec_id | packet_id |          type_id           | version |  state_id   | outgoing_flag
---------------+--------------+-----------+----------------------------+---------+-------------+---------------
           326 |          118 |         1 | request_project_create     | 1.0     | completed   |             0
           328 |          118 |         1 | notify_project_create      | 1.0     | completed   |             1
           329 |          118 |         2 | data_project_create        | 1.0     | completed   |             0
           330 |          118 |         2 | inform_transaction_complete | 1.0     | completed   |             1
           327 |          119 |         1 | request_account_create     | 1.0     | completed   |             0
           331 |          119 |         1 | notify_account_create      | 1.0     | completed   |             1
           332 |          119 |         2 | data_account_create        | 1.0     | completed   |             0
           333 |          119 |         2 | inform_transaction_complete | 1.0     | completed   |             1
           334 |          120 |         1 | notify_project_usage       | 1.0     | completed   |             1
           335 |          120 |         1 | inform_transaction_complete | 1.0     | in-progress |             0
           336 |          121 |        60 | request_user_modify        | 1.0     | completed   |             0
           337 |          121 |         1 | inform_transaction_complete | 1.0     | completed   |             1
           338 |          122 |        61 | request_user_modify        | 1.0     | completed   |             0
           339 |          122 |         1 | inform_transaction_complete | 1.0     | completed   |             1
           340 |          123 |        80 | request_account_inactivate | 1.0     | completed   |             0
           341 |          123 |         1 | notify_account_inactivate  | 1.0     | completed   |             1
           342 |          123 |        81 | inform_transaction_complete | 1.0     | in-progress |             0
           343 |          124 |        82 | request_project_inactivate | 1.0     | completed   |             0
           344 |          124 |         1 | notify_project_inactivate  | 1.0     | completed   |             1
           345 |          124 |        83 | inform_transaction_complete | 1.0     | in-progress |             0
(20 rows)

tgcdb
 packet_rec_id | trans_rec_id | packet_id |          type_id           | version | state_id  | outgoing_flag
---------------+--------------+-----------+----------------------------+---------+-----------+---------------
            15 |           13 |         1 | request_project_create     | 1.0     | completed |             1
            28 |           13 |         2 | data_project_create        | 1.0     | completed |             1
            26 |           24 |         1 | request_account_create     | 1.0     | completed |             1
            31 |           24 |         2 | data_account_create        | 1.0     | completed |             1
            34 |           25 |         1 | inform_transaction_complete | 1.0     | completed |             1
            35 |           26 |        60 | request_user_modify        | 1.0     | completed |             1
            37 |           27 |        61 | request_user_modify        | 1.0     | completed |             1
            39 |           28 |        80 | request_account_inactivate | 1.0     | completed |             1
            41 |           28 |        81 | inform_transaction_complete | 1.0     | completed |             1
            42 |           29 |        82 | request_project_inactivate | 1.0     | completed |             1
            44 |           29 |        83 | inform_transaction_complete | 1.0     | completed |             1
(11 rows)
```

Figure 1: Sample output from ddump.sh after successfully passing the test harness.