GWDG Archive Interface (gwrdifpk)

User and Reference Manual edition 1.0
Last updated February 27, 2014

Laurence D. Finston

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)

GWDG Archive Interface User and Reference Manual, edition 1.0.
The author is Laurence D. Finston.

Short Contents

Table of Contents

1 Introduction

`gwrdifpk` is a package that provides services for long-term archivation of files. It comprises the following components:

gwirdcli/gwrdsif
: A client/server application that provides remote access to an iRODS system and services pertaining to Handles and metadata.

gwirdpcl
: *Pull client*. A *dæmon* program that runs on the client side, accepting connections from the server program `gwirdsif`. `gwirdpcl` makes it possible for `gwirdsif` to actively request updates of archive objects. See Chapter 7 [Pull archiving], page 72.

gwrdwbap
: A web application that calls the client function `client_func`. See Section 26.1 [Main (and similar) functions], page 151. `gwrdwbap` is currently not being actively developed.

genpids
: A standalone program for generating handles a.k.a. *persistent identifiers* or PIDs.

gentans
: A standalone program for generating *transaction authentication numbers* or TANs. TANs could be used as an extra security measure, however, as of September 9, 2013, this feature, and hence '`gentans`', is not actively being developed.

Databases for Handles
: The Handle System uses a database for storing data. `gwrdifpk` uses two databases for this purpose, one for a "normal" Handle server that uses one or more prefixes registered with CNRI, another for a standalone Handle server with arbitrary prefixes. See Section 1.2 [Handles], page 2, Chapter 3 [Standalone handle service], page 9, and Chapter 27 [handlesystem and handlesystem_standalone Databases], page 158.

Databases for other data
: The `gwirdsif` and `gwirdcli` databases contain tables for other data used by the corresponding programs. See Chapter 28 [gwirdsif Database], page 160, and Chapter 29 [gwirdcli Database], page 174.

1.1 iRODS

iRODS provides a unified front-end that can be used for disparate archiving systems. See `https://www.irods.org`.

The iRODS server is programmed in C, however, it does not provide a C API. Instead, it provides two Java APIs, Jargon Core and Jargon Trunk, whereby the developers plan to phase out Jargon Trunk in favor of Jargon Core. In addition, it provides *icommands*, which are short C programs intended to be called from a shell.

For the purposes of `gwrdifpk`, it would be ideal if iRODS provided a C (or C++) API and this is what I would normally expect from a package written in C. It is possible to extract the assignments, function calls, etc., from the source code of the icommands and use them directly in one's own C (or C++) code. This requires a few minor changes in the

source code of the icommands. The author has tested this and it works. However, there is certain amount of work involved for each icommand, which would in sum be considerable. In addition, it would be necessary to merge the changes into any new release of iRODS and test them, whereby there is no guarantee that they would be compatible with the new version.

Therefore, in the current version of `gwrdifpk`, the icommands are called in pipes (using '`popen`'). Clearly, this is not ideal, but I doubt whether calling Java methods via JNI (Java Native Interface) would provide a significant performance advantage.

1.2 Handles

> "The Handle System, developed by Corporation for National Research Initiatives (CNRI), is an infrastructure on which applications serving many different purposes have been built. Among the objects we know of that are identified by handles are journal articles, technical reports, books, theses and dissertations, government documents, metadata, distributed learning content, and data sets. Handles are being used in digital watermarking applications, GRID applications, and repositories, registries and more."
> `http://www.handle.net/factsheet.html`

Handles are a form of *globally unique persistent identifiers* or *PIDs*. For example, '`0.NA/11858`' and '`11858/00-ZZZZ-0000-0001-6D1D-0`' are handles. Identifiers can only be unique within a domain. CNRI administers a decentralized infrastructure that defines the domain within which the handles are unique. Other institutions, such as the GWDG, may apply for *handle prefixes*, which are subject to a fee. In the examples above, the strings preceding the slash are the prefixes, i.e., '`0.NA`' and '`11858`'.

All handles have a prefix. The institution assigned a prefix is responsible for ensuring that all handles using that prefix are unique. Since no institution may use a prefix assigned to another, all handles are guaranteed to be unique with the domain comprising CNRI and the institutions to which prefixes have been assigned.

An important feature of the Handle System is the ability to *resolve* handles. That is, when a *handle client* requests resolution of a handle by submitting it to a handle server, the latter should return the *handle values* associated with the handle. If the server is responsible for the handle's prefix, it extracts the data from its own *handle database*. Otherwise, it may pass on the request to the handle server responsible for that prefix.

However, it is possible to set up a "standalone" handle server, i.e., one outside the domain administered by CNRI with arbitrary prefixes that may or may not duplicate ones used by other handle services. In this case, only "internal" handles may be resolved. See Chapter 3 [Standalone handle service], page 9.

`gwrdifpk` includes two handle databases: '`handlesystem`' for use with a "normal" handle server with an official prefix assigned by CNRI and '`handlesystem_standalone`' for use with a standalone handle server. The table definitions in the two databases are exactly the same. See Chapter 27 [handlesystem and handlesystem_standalone Databases], page 158.

1.3 gwrdifpk

`gwrdifpk` is a client/server application comprising the client program `gwirdcli` and the server program `gwirdsif`. They are written in C++.

The server program `gwirdsif` runs in the background (in normal use) and listens for *TLS* connections from the client program `gwirdcli`. The user calls the client, passing commands to it, which it sends to the server after *authentication and authorization* is performed using *X.509 certificates*.

The server processes the commands sent by the client and sends back responses, which may include commands for the client to execute. If so, a dialogue ensues which continues until both *peers* are finished, at which time the connection is closed.

See Chapter 4 [Getting Started], page 15, for more information on using `gwirdsif` and `gwirdcli`.

1.4 Source code and CWEB

The source code for `gwrdifpk` is written using Donald Knuth and Silvio Levy's *literate programming* package CWEB. See Section 2.2 [Prerequisites], page 4. Among other features, the `cweave` command from the CWEB package generates a "*pretty-printed*" version of the source code in the form of a TEX file, which can then be further processed to produce output in the formats DVI, PostScript and/or PDF.

Invoking `make` or `make all` in a shell in the top-level directory of the distribution, or `make`, `make all` or `make pdf` in the '`src`' subdirectory, will cause `cweave` to be called on various "driver" files to generated pretty-printed output for the programs included in the `gwrdifpk` package.

`cweave` formats source code and comments using TEX and creates section numbers, a table of contents, an index, a listing of named sections and cross-references. While it is very nice to have all of these things, for actually reading and working with the source code, it may be better to work with the original source files than with the `cweave` output. For one thing, the default formatting is not ideally suited to C++, especially with respect to input or output using the operators `>>` and `<<`. CWEB does provide ways to adjust the formatting, but the author has not yet found the time to work on this issue.

However, every effort has been made to ensure that the original source files are readable: The code is written in an "open" manner with many blank lines between statements and care has been taken with respect to indenting. In addition, the author uses Emacs' "`outline-minor-mode`" and the files are divided hierarchically into sections with headings of the form

```
@q * (1) @>
@q ** (2) @>
@q *** (3) @>
@q **** (4) @>
```

and so on. Readers can then use Emacs' functions for navigating from heading to heading, e.g., `outline-next-heading`, `outline-previous-heading`, `outline-up-heading`, etc. See Section "Outline Mode" in *The GNU Emacs Manual*.

To make it easier to work with CWEB files, the author has programmed a *CWEB mode* for GNU Emacs. The `gwrdifpk` distribution includes the necessary files of Emacs-Lisp code in the '`elisp`' subdirectory. See Chapter 34 [Emacs-Lisp files], page 192.

Please note, however, that GNU Emacs is *not* a prerequisite for working with the source files! Any plain text editor will do.

2 Installation

2.1 Obtaining the package

The GWDG Archive Interface may be obtained from `https://github.com`:

```
git clone https://github.com/gwdg/gwrdifpk.git
```
or
```
git clone git@github.com:gwdg/gwrdifpk.git
```

2.2 Prerequisites

iRODS `https://www.irods.org`

> "iRODS, which stands for i Rule Oriented Data Systems, is a project for building the next generation data management cyberinfrastructure." See Section 1.1 [iRODS], page 1.

Handle System
`http://www.handle.net/`

> "The Handle System provides efficient, extensible, and secure resolution services for unique and persistent identifiers of digital objects, and is a component of CNRI's Digital Object Architecture." See Section 1.2 [Handles], page 2.

CWEB `http://www-cs-faculty.stanford.edu/~knuth/cweb.html`

> "WEB is a software system that facilitates the creation of readable programs. [...] CWEB is a version of WEB for documenting C, C++, and Java programs."

CWEB should be included in most TEX distributions, though possibly as part of an "extended" version, as opposed to a basic version containing only the most commonly used TEX-related packages.

MySQL `http://www.mysql.com`
Database server and client library.

GnuTLS `http://www.gnutls.org`

> "GnuTLS is a secure communications library implementing the SSL, TLS and DTLS protocols and technologies around them. It provides a simple C language application programming interface (API) to access the secure communications protocols as well as APIs to parse and write X.509, PKCS #12, OpenPGP and other required structures. It is aimed to be portable and efficient with focus on security and interoperability."

GnuTLS, in turn, has a number of prerequisites of its own. Please see the GnuTLS documentation for more information. Please note that `gwrdifpk` uses GnuTLS version 2.4.1! Unfortunately, some functions from this version that `gwrdifpk` uses have been removed from more recent versions (3.x) of the GnuTLS library. An upgrade is planned.

GNU Privacy Guard (GnuPG)
: `http://www.gnupg.de`

  "GnuPG is the GNU project's complete and free implementation of the OpenPGP standard as defined by RFC4880 . GnuPG allows to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. Version 2 of GnuPG also provides support for S/MIME."

expat
: `http://expat.sourceforge.net`

  "Expat is an XML parser library written in C. It is a stream-oriented parser in which an application registers handlers for things the parser might find in the XML document (like start tags)."

FastCGI
: `http://www.fastcgi.com/drupal`

  "The Fast Common Gateway Interface (FastCGI) is an enhancement to the existing CGI (Common Gateway Interface), which is a standard for interfacing external applications with Web servers."

Flex
: `http://flex.sourceforge.net/`

  "Flex is a tool for generating scanners. A scanner, sometimes called a tokenizer, is a program which recognizes lexical patterns in text."

GNU Bison
: `http://www.gnu.org/software/bison/`

  "Bison is a general-purpose parser generator that converts an annotated context-free grammar into a deterministic LR or generalized LR (GLR) parser employing LALR(1) parser tables."

GNU Make
: `http://www.gnu.org/software/make`

  "[. . .] the GNU 'make' utility, [. . .] determines automatically which pieces of a large program need to be recompiled, and issues the commands to recompile them."

  Make will almost certainly already be installed on any GNU/Linux or other Unix-like system.

All of the files needed for building the package are included in the GitHub repository (see Section 2.1 [Obtaining the package], page 4) and the distribution generated from '`make dist`'. If, however, a user wishes to build the package "from scratch", there are additional prerequisites:

Libtool
: `http://www.gnu.org/software/libtool`

Autoconf
: `http://www.gnu.org/software/autoconf`

Automake
: `http://www.gnu.org/software/automake`

Texinfo
: `http://www.gnu.org/software/texinfo` For generating this manual.

Most, if not all, of the prerequisites for `gwrdifpk` can be installed by using the package managers supplied with common GNU/Linux distributions.

2.3 Building

The `gwrdifpk` distribution includes a '`configure`' script, '`Makefile.in`' and all other required files, so the package can be built by simply invoking '`configure`' followed by '`make all`' and '`make install`'. However, the '`configure.ac`' and '`Makefile.am`' files are also included in the distribution, so that the package may be built "from scratch", if desired:

```
libtoolize && aclocal && autoconf && autoheader && \
   automake --add-missing --copy
```

The options '`--add-missing`' and '`--copy`' to '`automake`' only need to be used the first time '`automake`' is called.

`gwrdifpk` does not require *root privileges*. On the other hand, the default installation directory of the '`configure`' script (as generated by Autoconf) is '`/usr/local/bin`', which will normally belong to '`root`'. Therefore, when installing `gwrdifpk` as a user without root privileges, it will be necessary to specify a different installation directory with the '`--prefix`' option:

```
./configure LIBS="-pthread -lm -lgnutls -lgcrypt -lexpat" \
   --prefix=‘pwd‘
```

By default, '`make`' will create *shared libraries* when building the package. This is good for production versions, but not so good for testing purposes, because it's time-consuming. The '`--disable-shared`' option can be used to suppress building shared libraries:

```
./configure LIBS="-pthread -lm -lgnutls -lgcrypt -lexpat" \
   --prefix=‘pwd‘ --disable-shared
```

If some header files or libraries needed by `gwrdifpk` `gwrdifpk` are not in locations where they will found by the system "automatically", for example, on GNU/Linux systems, if the library directories are not listed in the file '`/etc/ld.so.conf`' or included in the environment variable '`LDFLAGS`', they will have to be passed to '`configure`' specially. For example, if some required header is located in '`/usr/users/lfinsto/my_header_dir`' and a library in '`/usr/users/lfinsto/my_library_dir`', '`configure`' could be invoked like this:

```
./configure CPPFLAGS="-I/usr/users/lfinsto/my_header_dir" \
    LDFLAGS="-L/usr/users/lfinsto/my_library_dir" \
    LIBS="-pthread -lm -lgnutls -lgcrypt -lexpat" \
   --prefix=‘pwd‘ --disable-shared
```

The shellscript '`pcfinston_master_config.sh`' in the top-level directory (i.e., '`gwrdifpk-1.0/`') contains an example of how one could invoke '`configure`'.

After '`configure`' and '`make`' succeed, the package may be used. If desired, '`make all`' and '`make install`' may be run, too. '`make install`' causes the programs and libraries to be installed in the default locations or in those specified with the options to '`configure`' (see above). The server program `gwirdsif` may be started in a shell and the client program `gwirdcli` may be invoked in another shell to communicate with it.

2.4 X.509 certificates

`gwrdifpk` uses *X.509 certificates* for *authorization* and *authentication*. For production use, "genuine" certificates, i.e., certificates issued by a recognized *certification authority* (*CA*) must be used. However, for testing purposes certificates may be generated ad hoc by using `certtool` (see Section "Invoking certtool" in *GnuTLS*). or `openssl`.

`gwrdifpk` includes the shellscript '`[...]/src/gen_x509_cert_key_pair.sh`', which makes it easier to generate a certificate-key pair. See See Section 33.2 [X.509 certificates], page 191. In addition, `gen_x509_cert_key_pair.sh` calls `certtool --certificate-info ...` to create a file containing the information from the certificate in human-readable form.

The *distinguished name* or (DN) from a user's certificate must be entered into the row for that user in the '`gwirdsif.Users`' database table. The distinguished name is based on the '`Subject`' field of the certificate. For example, if a certificate-key pair is generated for a person named "John Smith" with username '`jsmith`', the '`Subject`' field in his certificate (translated into human-readable form) might look like this:

```
Subject: C=DE,O=GWDG,OU=gwrdifpk,L=Goettingen,ST=Germany,CN=John Smith,UID=2
```

so that his distinguished name will look like this:

```
/C=DE/ST=Germany/L=Goettingen/O=GWDG/OU=gwrdifpk/CN=John Smith
```

'`CN`' stands for "Common Name". In the distinguished name, the codes for the fields, i.e., '`C`' for "Country", '`O`' for "Organization", etc., are preceded by slashes, the commas have been removed, as has the '`UID`' field.

2.5 Database setup

`gwrdifpk` uses three databases, `handlesystem` or `handlesystem_standalone`, `gwirdsif` and `gwirdcli`. The first two are used by the server program `gwirdsif` only, while the third is used by the client program `gwirdcli` only. Of the three, `gwirdsif` contains the most tables.

The auxiliary program `setupds` can be used to set up the database tables used by `gwrdifpk`. See Section 32.5 [Set up databases], page 185, for instructions on invoking `setupdbs`.

2.5.1 Setting up iRODS users

iRODS users may be created as described in the iRODS documentation. That is, an iRODS administrator uses the `mkuser` command in the `iadmin` environment to create a user and sets his or her password using the `moduser` command. The iRODS username must correspond to the `gwrdifpk` username.

It's most convenient to create *iRODS environment files* in '`$HOME/.irods/`' and to call `iinit` to create files containing the "scrambled" iRODS passwords.

An iRODS environment file for a user '`abrown`' in '`$HOME/.irods/`' might be named '`.irodsEnv.abrown`' and have the following contents:

```
# iRODS personal configuration file.
#
# iRODS server host name:
irodsHost 'pcfinston.gwdg.de'
```

```
# iRODS server port number:
irodsPort 1247

# Default storage resource name:
irodsDefResource 'demoResc'
# Home directory in iRODS:
irodsHome '/tempZone/home/abrown'
# Current directory in iRODS:
irodsCwd '/tempZone/home/abrown'
# Account name:
irodsUserName 'abrown'
# Zone:
irodsZone 'tempZone'
irodsAuthFileName '/home/lfinsto/.irods/.irodsA.abrown'
```

The last line indicates that the scrambled iRODS password should be stored in '`/home/lfinsto/.irods/.irodsA.abrown`'. If I copy '`.irodsEnv.abrown`' to '`.irodsEnv`' or set the '`irodsEnvFile`' environment variable,

```
export irodsEnvFile=/home/lfinsto/.irods/.irodsEnv.abrown
```

and call `iinit`, '`.irodsEnv.abrown`' will indeed be created.

On my system, the scrambled password is only 14 characters long. It is does not provide any real security, so it is really more of a nuisance than anything else. However, at the present time, the iRODS passwords are needed in `gwrdifpk`, so they are encrypted using GPG (the GNU Privacy Guard) stored in the '`gwirdsif.Users`' database table. Since the "scrambling" algorithm obviously uses the timestamp of the file containing the scrambled password, the timestamp is stored in the table as well.

To encrypt the iRODS passwords, and decrypt them later, a GPG key pair is needed. On my system, it's name is "GWDG iRODS Interface Server (gwirdsif) <lfinsto@gwdg.de>", however any other name may be used. It is, however, important at the present time that the secret key *not* be protected by a passphrase: The reason is that `gwirdsif` must run unattended and so it's not possible to have someone type in a passphrase everytime an iRODS password is needed. See See ⟨undefined⟩ [Using the GNU Privacy Guard], page ⟨undefined⟩. At some time, the author may change this so that the passphrase for the GPG secret key is entered once when `gwirdsif` is invoked and saved as safely as possible until the program terminates.

To make it easier to write the encrypted, scrambled iRODS password and the timestamp used when it was scrambled to the database table, `gwrdifpk` includes the shellscript `update_irods_passwd.sh`. See Section 33.3 [iRODS passwords], page 191.

3 Standalone handle service

3.1 Background

Normally, a *handle service* is set up to use a prefix or prefixes registered with the Corporation for National Research Initiatives (CNRI) (`http://www.cnri.reston.va.us/`). In this case, the normal procedure for *resolving a handle* is for the *handle client* to first contact CNRI's *global handle service* in order to retrieve the *service information* for a given prefix. For example, to resolve the handle '`11858/00-DEMO-0000-0000-0012-8`', the service information for the prefix '`11858`' must first be retrieved from CNRI's *Global Handle Registry* (GHR). This information is stored in the handle '`0.NA/11858`'. The handle client can now contact this service to resolve the handle '`11858/00-DEMO-0000-0000-0012-8`'.

However, it is possible to set up a *standalone handle service*, as described in the Handle System documentation (*HANDLE.NET (version 7.0) Technical Manual, Version 1.1*).

Please note: A standalone handle service should *only* be used for testing purposes! For production use, only prefixes registered with CNRI should be used. The author also strongly recommends against using a standalone handle service "internally": For this purpose, it would be better to use some other form of identifier and not handles because of the risk of confusion between "internal" and "official" handles.

The easiest way to set up a standalone handle service is to use an existing prefix assigned by the Global Handle Registry (GHR). The service information in the *naming authority handle* (e.g., '`0.NA/11858`', as above) in the GHR includes the IP address of the server responsible for resolving handles with that prefix. If a test server with a different IP address is used, the handles maintained by that server will not be resolved upon requests to the GHR. Nor will they be resolved upon requests to other servers, unless these are also specially configured to find the test server, i.e., to use the service information for the test server and not to use the GHR for resolving handles with that prefix. Otherwise, only requests to the test server itself will cause handles on that server to be resolved. Please note that in this case authorization *is* performed using global resolution. In other words, the authorization handle is stored in the GHR and not in the test handle server's local database.

3.2 Setup

The normal procedure for setting up a handle server is described in the file '`INSTALL.txt`', which is included in the handle system distribution. After running '`hdl-setup-server`', one is instructed to send the generated file '`sitebndl.zip`' to the CNRI Handle System Administrator (`hdladmin@cnri.reston.va.us`):

> "The Administrator will then create the prefix on the root service (known as the Global Handle Registry(GHR)), and notify you when this has been completed. You will not be able to continue the install until you receive further information concerning your prefix.
>
> ONCE YOU RECEIVE YOUR PREFIX INFORMATION FROM HDLADMIN THEN PROCEED WITH THE FOLLOWING STEPS TO 'HOME' YOUR PREFIX TO YOUR NEW SERVICE."

Please note that it is necessary to follow the instructions in Chapter 10 of the *HANDLE.NET Technical Manual* regarding the files '`local_nas`' and '`resolver_site`' in the

user's home directory and the file '`config.dct`' in the directory containing the server configuration (e.g., '`/home/my_account/hs/svr_1`'). The user that is meant is the one under whose account Handle Service *clients* are invoked, e.g., '`hdl-admintool`', '`hdl-dbtool`', etc. However, despite what is said in the *HANDLE.NET Technical Manual*, a prefix of the form '`0.NA/<prefix>`', e.g., '`0.NA/12345`' can be used with local resolution; it is not necessary to create a different handle for administration, e.g., '`12345/ADMIN`'.

3.3 Database Entries

The main difficulty in setting up a standalone handle server is creating the database entries for the *Naming Authority handle*. Normally, this is done by the CNRI Handle System Administrator and authentication is performed using *global resolution* via the *Global Handle Registry* (GHR).

A standalone handle server, on the other hand, cannot use the GHR to resolve handles and therefore the local database must contain the entries for the naming authority handle of the standalone handle server. It would, of course, be possible to create the database entries by hand using SQL commands, but this would require deeper knowledge of how the data must be formatted. Practically speaking, this isn't possible for most people, especially when they are just starting out using the handle system in the first place.

The easiest way to obtain this knowledge is to apply for a temporary prefix for testing. In this case, one can use global resolution to display the database entries for the naming authority prefix stored at the Global Handle Registry *and copy them* to the local database using '`hdl-admintool`'! Now, one may run a standalone handle server with local resolution using this prefix, and/or copy and modify the database entries in order to use them with any number of other prefixes.

For example, if my prefix is 00001, I can use '`hdl-admintool`' to view the handle '`0.NA/00001`' containing the service information for my handle server. I can then use the "Copy Values" and "Create Handle" buttons to create a new handle, e.g., '`00001/00001-COPY`'. Please note that I must use '`00001`' as the prefix, because I am only authorized to create handles under this prefix and in particular not under the global prefix '`0.NA`'.

Now, I can query my local database for the entries that were created. I assume the use of an SQL database rather than the "built-in" database, as described in Chapter 7 of the *HANDLE.NET Technical Manual*. Please note that the JAR file for the MySQL JDBC connector, or a symbolic link to it, must be present in the '`lib`' directory of the handle server installation, e.g., '`/home/my_account/hs/hsj-7.1/lib`'. Otherwise, the handle server will not be able to connect to the database.

```
mysql> select * from handles where handle = '00001/00001-COPY'\G
⇒
*************************** 1. row ***************************
     handle: 00001/00001-COPY
        idx: 1
       type: HS_SITE
       data: [binary data]
   ttl_type: 0
        ttl: 86400
```

```
  timestamp: 1346335399
       refs:
 admin_read: 1
admin_write: 1
   pub_read: 1
  pub_write: 0
*************************** 2. row ***************************
     handle: 00001/00001-COPY
        idx: 2
       type: EMAIL
       data: laurence.finston@gwdg.de
[...]
*************************** 3. row ***************************
     handle: 00001/00001-COPY
        idx: 3
       type: DESC
       data: Test prefix for GWDG - JHE - 8/28/12
[...]
*************************** 4. row ***************************
     handle: 00001/00001-COPY
        idx: 100
       type: HS_ADMIN
       data: [binary data]
[...]
*************************** 5. row ***************************
     handle: 00001/00001-COPY
        idx: 101
       type: HS_ADMIN
       data: [binary data]
[...]
*************************** 6. row ***************************
     handle: 00001/00001-COPY
        idx: 200
       type: HS_VLIST
       data: [binary data]
[...]
*************************** 7. row ***************************
     handle: 00001/00001-COPY
        idx: 300
       type: HS_PUBKEY
       data: [binary data]
[...]
7 rows in set (0.00 sec)
```

The next step is to add the files '`local_nas`' and '`resolver_site`' in the '`.handle/`' directory directly below the user's home directory, as mentioned above, in order to "turn off" global resolution. Then, the value of the '`handle`' field in these entries must be changed.

I could use '00001' as my prefix, but I don't have to. Let's say I want to use '12345' instead, so I change the value of the '**handle**' column in these entries from '00001/00001-COPY' to '0.NA/12345':

```
mysql> update handles set handle = '0.NA/12345'
          where handle = '00001/00001-COPY';
⇒
Query OK, 7 rows affected (0.00 sec)
Rows matched: 7  Changed: 7  Warnings: 0
```

3.3.1 Homing a Prefix

The explanation of homing a prefix in Chapter 10 of the *HANDLE.NET Technical Manual* is unfortunately not very clear. It explains that a server must be "told" that it's responsible for resolving certain prefixes, but it doesn't explain exactly what this entails. In fact, a prefix is "homed" if the database table '**nas**' contains an entry for it:

```
mysql> select * from nas;
⇒
+------------+
| na         |
+------------+
| 0.NA/0.NA  |
| 0.NA/00001 |
| 0.NA/12345 |
| 0.NA/55555 |
+------------+
4 rows in set (0.00 sec)
```

Please note that the prefix '0.NA/0.NA' is also homed. This makes it possible to resolve the handles with the prefix '0.NA', i.e., '0.NA/00001', '0.NA/12345' and '0.NA/55555' locally, i.e., without contacting the GHR.

3.3.2 Replacing and Modifying the Binary Data

Five of the seven database entries for our naming authority handle '0.NA/12345' contain binary data, which may need to be modified:

```
 idx: 1
type: HS_SITE

 idx: 100
type: HS_ADMIN

 idx: 101
type: HS_ADMIN

 idx: 200
type: HS_VLIST

 idx: 300
type: HS_PUBKEY
```

The data for 'HS_SITE' is simply the contents of the file 'siteinfo.bin' in the directory containing my server configuration. For this example, there is therefore no need to change it. However, for a different site, one just has to replace it with the contents of the appropriate 'siteinfo.bin' file. However, the MySQL function 'load_file' requires (among other things) that the file be readable by all. I therefore copy the contents to another file before calling 'update', e.g.:

```
cd
cp hs/svr_1/siteinfo.bin ttemp.txt
chmod a+r ttemp.txt
update handles set data = load_file('/home/my_account/ttemp.txt')
   where handle = '0.NA/12345' and type = 'HS_SITE';
```

Here, it does matter that 'load_file' adds a newline to the end of the data from the file.

The Handle Proxy made available by CNRI at http://hdl.handle.net displays the binary data for handle values with 'type' 'HS_ADMIN' and 'HS_VLIST' in a human-readable form. The following are the values for the original handle '0.NA/00001', which we've copied:

```
Index   Type      Data
100     HS_ADMIN handle=0.NA/0.NA; index=200; [create hdl,delete hdl,\
                 read val,modify val,del val,add val,\
                 modify admin,del admin,add admin]

101     HS_ADMIN handle=0.NA/00001; index=200; \
                 [create hdl,read val,modify val,add val,list]

200     HS_VLIST 300:0.NA/00001
```

We don't have to worry about the handle value with index = 100. It refers to an administrator of the global handle system.

When simply selecting the data fields from the database, they are (mostly) unreadable:

```
mysql> select data from handles where handle = '00001/00001COPY'
         and (idx = 101 or idx = 200);
⇒
select data from handles where handle = '00001/00001COPY'
    and (idx = 101 or idx = 200);
+------------------------+
| data                   |
+------------------------+
| ^LQ
0.NA/00001   \310    |
|    ^A
0.NA/00001  ^A, |
+------------------------+
2 rows in set (0.00 sec)
```

However, the name of the handle is obviously stored in plain-text. It works to just replace it with the name we want:

```
mysql> set @a = (select replace ((select data from handles
```

```
                                where handle = '0.NA/12345' and idx = 101),
                               '00001', '12345'));
mysql> update handles set data = @a
           where handle = '0.NA/12345' and idx = 101;
```

It is necessary to use the user-defined variable '`@a`' to store the result of the call to the '`replace`' function, because the target table of the '`update`' command cannot appear in a subquery.

Of course, this only works if one has authorization to copy an existing naming authority from the GHR in the first place.

Now, the handle server must be restarted and, with a bit of luck, you'll have a standalone handle server with local resolution!

4 Getting Started

4.1 gwirdsif

The server program must "know" the location of the iRODS server in order to function. If it's stored in the *environment variable* `IRODS_SERVER_DIR`, e.g.,

```
export IRODS_SERVER_DIR=/home/lfinsto/iRODS
```

then starting the server program `gwirdsif` can be as simple as this:

```
gwirdsif
```

Alternatively, the location may be passed to `gwirdsif` using the *command-line option* '`--irods-server-directory`':

```
gwirdsif --irods-server-directory /home/lfinsto/iRODS
```

`gwirdsif` takes many more options, but they have sensible defaults. For more information on `gwirdsif's` options, see Chapter 6 [Invoking gwirdsif/gwirdcli], page 63.

When `gwirdsif` is started in a shell, it prints some information for the user to the terminal and then waits for connections from the client:

```
gwirdsif --irods-server-directory /home/lfinsto/iRODS
⇒
# 1377700507  'Wed, 28 Aug 2013 15:35:07 GMT'
Started run
iRODS server directory:  /home/lfinsto/iRODS
iRODS server running.  PID:  27237
Process ID:     12985
Socket path:    /tmp/gwirdsif.sock
Log directory:  /home/lfinsto/.gwirdsif
# 1377700507  'Wed, 28 Aug 2013 15:35:07 GMT'
Started run
[Thread 3] In 'purge_irods_archive':  Deleted 0 iRODS objects \
   from archive.
Deleted 0 iRODS objects from 'gwirdsif' database.
[Thread 4]  In 'listen_local':         Server ready. \
   Listening to Unix domain socket '/tmp/gwirdsif.sock'.
[Thread 6]  In 'listen_remote_X_509':  Server ready. \
   Listening to port 5557.
[Thread 5]  In 'listen_remote_anon':   Server ready. \
   Listening to port 5556.
```

`gwirdsif` "listens" for connections at three different places: The most important is port 5557 (by default), because authentication/authorization using X.509 certificates is performed for TLS connections using this port. This is the only kind of connection that should be used in a production environment; the others are for testing purposes *only*! '`/tmp/gwirdsif.sock`' is a Unix domain socket and can therefore only be used for local

connections while port 5556 (again, by default), like port 5557, is used for TLS connections, but without any authentication/authorization.[1]

Normally, it is intended that `gwirdsif` be run as a *dæmon process*, i.e., it runs in the background and doesn't terminate when the user who started it logs out. For example, it could be invoked like this:

```
nohup gwirdsif > /home/lfinsto/.gwirdsif/gwirdsif.stdout \
   2> /home/lfinsto/.gwirdsif/gwirdsif.stderr &
```

Here, `gwirdsif's` output to standard output and standard error is *redirected* to the files '`/home/lfinsto/.gwirdsif/gwirdsif.stdout`' and '`/home/lfinsto/.gwirdsif/gwirdsif.stderr`'. The directory '`$HOME/.gwirdsif/`' is the default for `gwirdsif`'s log directory, which may also be set using the '`log-directory`' option or by setting the environment variable `GWIRDSIF_DIR`.

If the output is redirected to files named '`gwirdsif.stdout`' and '`gwirdsif.stderr`' in the log directory, the function `purge_server_logs` will take care of rotating them. If other paths are chosen, the files will not be rotated. See Section 26.10 [Deleting and rotating files], page 154.

For testing purposes, however, it's much more useful to invoke `gwirdsif` as a foreground process (and without `nohup`), so that its output is written to the terminal.

4.2 gwirdcli

4.2.1 Invoking gwirdcli

The simplest way to invoke `gwirdcli` is with a single argument, namely the hostname of the machine where the server program `gwirdsif` is running. For example, if `gwirdsif` is running on '`pcfinston.gwdg.de`', `gwirdcli` may be invoked like this:

```
gwirdcli pcfinston.gwdg.de
⇒
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
whoami
.
⇒
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston
```

[1] The ports used can be reset by means of the command-line options '`--x509-port`' and '`--anon-port`', respectively. See Section 6.2 [Command-line options], page 63. Of course, if the ports are changed, they must be changed for both the client and the server!

When invoked in this way, `gwirdcli` prompts the user for commands which it will send to the server. The server processes them and sends a *response* to the client, which prints out a message to the terminal (or whatever `gwirdcli`'s *standard output* happens to be connected to).

In this example, something important happened "behind the scenes", namely authentication/authorization with X.509 certificates. The default filenames for the user's certificate and key are '`user_cert.pem`' and '`user_key.pem`', respectively. If these aren't the names of the user's certificate and key (or of symbolic links to them), then `gwirdcli` will have to be invoked using the '`--cert-filename`' and '`--key-filename`' options:

```
gwirdcli --cert-filename my_cert.pem --key-filename my_key.pem \
   pcfinston.gwdg.de
⇒
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.
[...]
```

If the server has been installed on the localhost, then the hostname argument can be left off:

```
gwirdcli
⇒
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
whoami
.
⇒
Authentication error -->
Error code:  1
Exiting.
Unauthenticated connection and "DISTINGUISHED_NAME" command was either \
   not sent to server, or failed.
Please note that unauthenticated connections are only for \
   testing purposes!
Exiting.
```

Oops! In this case, the client connects with the server via the Unix Domain socket '`/tmp/gwirdsif.sock`', so that authentication/authorization using X.509 certificates is not performed, which is only allowed for testing purposes. In this case, the user must provide a *distinguished name* to identify himself or herself to the server:

```
gwirdcli
⇒
Enter commands:
```

```
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
distinguished_name \
"/C=DE/ST=Germany/L=Goettingen/O=GWDG/OU=gwrdifpk/CN=Laurence Finston"
get_user_info
.
⇒
Get user info response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
distinguished_name:
     organization:.............GWDG
     organizationalUnitName:.....gwrdifpk
     commonName:................Laurence Finston
     countryName:...............DE
     localityName:..............Goettingen
     stateOrProvinceName:.......Niedersachsen
     user_id:...................1
     user_name:.................lfinsto
[...]
```

Of course, there's nothing to prevent the user from sending some other user's distinguished name to the server, so that this feature is only for testing purposes.

In the examples so far, `gwirdcli` has exited immediately after sending a single batch of commands to the server, receiving its responses and printing them to the terminal. Often, however, the user will want to have a dialogue with the server. The option '`--no-terminate-on-end-input`' can be used for this purpose:

```
gwirdcli --no-terminate-on-end-input localhost
⇒
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
whoami
.
⇒
Whoami response -->
Response code:  0
User Info:
user_id:      1
```

```
username:     lfinsto
Common Name:  Laurence Finston

Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.
```

(User input:)

```
show groups all
.
⇒
Show groups response -->
Response code:  0
Group info for 2 groups:
Group_Type:
group_id ==           1
'group_name' ==       test_group_0
'creator_id' ==       1
'creator_username' == lfinsto
'created' ==          1370433954 == 2013-06-05 14:05:54
[...]
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.
```

(User input:)

```
q
⇒
Exiting.
```

In this example, '`localhost`' is the server hostname argument. In this case, a TLS connection with X.509 authentication/authorization is used, so that the `distinguished_name` command isn't needed.

Users don't have to type in commands at a prompt, however. Another way of passing intput to `gwirdcli` is to use a pipe:

```
echo "whoami" | gwirdcli localhost
⇒
Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.
Whoami response -->
Response code:  0
User Info:
user_id:      1
```

```
username:     lfinsto
Common Name:  Laurence Finston
```

The prompt is printed to the terminal even though input has already been provided. It can be suppressed using the '`--suppress-prompt`' option:

```
echo "whoami" | gwirdcli --suppress-prompt localhost
⇒
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston
```

Typing in commands at a prompt or passing a couple of commands to `gwirdcli` via a pipe may be useful sometimes, but in most cases, it will be more convenient to put the commands into a file and pass the filename to `gwirdcli`.

The file can be passed using *redirection*:

```
cat sample_input.txt
⇒
whoami

gwirdcli --suppress-prompt localhost < sample_input.txt
⇒
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston
```

Alternatively, `gwirdcli` can be invoked with the '`--input-filename`' option:

```
gwirdcli localhost --input-filename sample_input.txt
⇒
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston
```

In this case, no prompt is printed to the terminal.

If the '`--no-terminate-on-end-input`' option is used, then the user can have a dialogue with the server after the commands in the input file have been processed:

```
gwirdcli localhost --input-filename sample_input.txt \
   --no-terminate-on-end-input
⇒
Whoami response -->
Response code:  0
```

```
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston

Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
ls
.
⇒
ls -->
/tempZone/home/lfinsto:
  abc.txt

Enter commands:
Type commands followed by <ENTER>.  Multiple lines may be entered.
Enter a single period ('.') on a line to finish.
Use 'q' (or 'Q') command to quit.

(User input:)
q
⇒
Exiting.
```

Please note that a dialogue is only possible when the user types the first batch of commands after the prompt or a file is specified using the '`--input-filename`' option. That is, it is not possible when passing input to `gwirdcli` using a pipe or redirection. In these cases, standard input has been disconnected from the terminal so the latter can no longer be used for passing input to `gwirdcli`.

4.2.2 Putting and Getting iRODS Objects

Let's say I have a file '`abc.txt`' that I want to send to the server and have it stored in the remote iRODS archive. The command for this is `put`:

```
echo "put abc.txt" | gwirdcli --suppress_prompt localhost
⇒
put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     'iput' command succeeded, returning 0
```

Lets's say now I've deleted my local copy of '`abc.txt`' and I want to restore it from the remote iRODS archive. The command for this is `get`:

```
ls -l abc.txt
⇒
```

```
ls: cannot access abc.txt: No such file or directory

(User input:)
echo "get abc.txt" | gwirdcli --suppress-prompt localhost
⇒
get -->
Local filename:  abc.txt
Response code:   2
Response:        Success.  Queuing "SEND FILE" command.

get -->
Remote filename: /tempZone/home/lfinsto/abc.txt
Local filename:  abc.txt
Exit status:     0
Overwrite:       False

Received remote file ‘/tempZone/home/lfinsto/abc.txt’.
Stored in local file ‘abc.txt’.

(User input:)
ls -l abc.txt
⇒
-rw-r----- 1 lfinsto users 5064 Aug 29 15:22 abc.txt
```

iRODS objects can have *Attribute-Value-Unit triples* (AVUs) associated with them. They can be shown by the `get metadata` command:

```
echo "get metadata abc.txt" | gwirdcli --suppress-prompt localhost
⇒
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  0

No user-defined metadata (AVUs) to display
```

‘`abc.txt`’ doesn’t have any AVUs because `put` was called without any options.

The GWDG Archive Interface uses *handles* to store information about iRODS objects and other entities. Normally, when a file is “put”, options are used to tell the server to generate a handle for it:

```
echo "put +pid +gen abc.txt" | gwirdcli --suppress-prompt localhost
⇒
put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  1
Response:     Server error:  ‘iput’ command failed, returning 3:
ERROR: putUtil: put error for /tempZone/home/lfinsto/abc.txt, \
   status = -312000 status = -312000 OVERWRITE_WITHOUT_FORCE_FLAG
```

Oops! ‘abc.txt’ already exists in the remote archive. We can use the ‘-f’ flag to tell the server to overwrite it:

```
echo "put -f +pid +gen abc.txt" | gwirdcli --suppress-prompt localhost
⇒
put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     ‘iput’ command succeeded, returning 0

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Success:  Generated PID ‘12345/00001’

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Added handle values with type == ‘IRODS_OBJECT’ \
   and type == ‘CREATOR_INDEX’ successfully

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Success:  Stored PID ‘12345/00001’ in \
   iRODS object metadata
```

Now, the remote iRODS object ‘/tempZone/home/lfinsto/abc.txt’ should have an AVU. In these examples, the server is running on the same host as the client, so I can use the normal *icommands* to access the iRODS server:

```
imeta ls -d abc.txt
⇒
AVUs defined for dataObj abc.txt:
attribute: PID
value: 12345/00001
units:
```

The output from the `get metadata` command contains this information, but also quite a bit more, some of which I’ve left out of the following example to reduce clutter:

```
echo "get metadata abc.txt" | gwirdcli --suppress-prompt localhost
⇒
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  1

Irods_Object_Type:
id ==                                    0
path ==                                  /tempZone/home/lfinsto/abc.txt
```

```
[...]
avu_vector.size() == 1
Showing avu_vector:
Irods_AVU_Type:
id ==                        0
irods_object_id ==           0
user_id ==                   0
irods_object_path ==         (empty)
attribute ==                 PID
value ==                     12345/00001
units ==                     (empty)
time_set ==                  1367603247 (seconds since epoch):  \
                                2013-05-03 19:47:27 CEST +0200
deleted_from_archive ==      0
deleted_from_gwirdsif_db == 0
[...]
```

The handle '12345/00001' which has been generated for the iRODS object 'abc.txt' can be retrieved by using the `get handle` command:

```
get handle pid "12345/00001"
⇒
get handle -->
Response code:                  0
filename:                       /tempZone/home/lfinsto/abc.txt
handle:                         12345/00001
idx:                            1
type:                           IRODS_OBJECT
data_length:                    30
data:                           /tempZone/home/lfinsto/abc.txt
ttl_type:                       0
ttl:                            86400
timestamp:                      1377784493 (2013-08-29 15:54:53 CEST)
refs_length:0
refs:                           NULL
admin_read:                     1
admin_write:                    1
pub_read:                       1
pub_write:                      0
handle_id:                      56
handle_value_id:                130
irods_object_id:                0
created:                        1377784493 (2013-08-29 15:54:53 CEST)
last_modified:                  0
delete_from_database_timestamp: 0
created_by_user:                1
marked_for_deletion:            0
```

```
get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            211
type:                           CREATOR
data_length:                    68
data: \
   /C=DE/O=GWDG/OU=gwrdifpk/L=Goettingen/ST=Germany/CN=Laurence Finston
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            300
type:                           HS_ADMIN
data_length:                    22
data:                           ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
```

The server generates a separate response for each handle value. The handle values are displayed on the terminal, but they are also stored in the client-side database '`gwirdcli`' in the '`handles`' table.

4.2.3 Dublin Core Metadata

`gwrdifpk` provides facilities for storing and retrieving *Dublin Core metadata*, i.e., XML data conforming to the standards developed in connection with the Dublin Core Metadata Initiative (DCMI): `http://dublincore.org`

Let's say I've '`put`' the file '`abc.txt`' and generated the handle '`12345/00001`', as in Section 4.2.2 [Putting and Getting iRODS Objects], page 21, and that I have some Dublin Core metadata in file '`metadata_sample_1.xml`':

```
cat metadata_sample_1.xml
⇒
<?xml version="1.0"?>

<metadata
  xmlns="http://example.org/myapp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/myapp/ \
     http://example.org/myapp/schema.xsd"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/">

  <dc:title xsi:type="title attribute" xsi:typex="title attribute 2">
    Sample Dublin Core Metadata (Title)
```

```
  </dc:title>
  <dc:creator>
    Laurence D. Finston (Creator)
  </dc:creator>
  <dc:subject>
    Sample Dublin Core Metadata 1 (Subject)
  </dc:subject>
  <dc:description>
    Sample Dublin Core Metadata 1 (Description)
  </dc:description>
  <dc:publisher>
    GWDG 1 (Publisher)
  </dc:publisher>
  <dc:contributor>
    Sample contributor 1
  </dc:contributor>
  <dc:date>
    2012-12-06 12:11:26
  </dc:date>
  <dc:type>
    iRODS object (Type)
  </dc:type>
  <dc:format>
    ASCII (Format)
  </dc:format>
  <dc:identifier>
    XXX (Identifier)
  </dc:identifier>
  <dc:source>
    GWDG (Source)
  </dc:source>
  <dc:language>
    English (Language)
  </dc:language>
  <dc:relation>
    Not applicable (Relation)
  </dc:relation>
  <dc:coverage>
    Not applicable (Coverage)
  </dc:coverage>
  <dc:rights>
    All rights reserved (Rights)
  </dc:rights>
  <dcterms:abstract>
    Sample Abstract
  </dcterms:abstract>
</metadata>
```

I can now send the contents of 'metadata_sample_1.xml' to the server and associate it with the (server-side) iRODS object 'abc.txt':

```
add metadata metadata_sample_1.xml abc.txt
⇒
add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle for metadata:  12345/00002.

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00001’ \
      with type ‘IRODS_OBJECT_PID’ to handle ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for iRODS object \
      ‘/tempZone/home/lfinsto/abc.txt’ with type ‘IRODS_OBJECT_REF’ \
      to handle ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00002’ \
      with type ‘DC_METADATA_PID’ to handle ‘12345/00001’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Call to ‘imeta’ succeeded.  Added AVU \
      with type ‘DC_METADATA_PID’ and value ‘12345/00002’ \
      to iRODS object ‘/tempZone/home/lfinsto/abc.txt’.

add metadata-->
```

```
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   (Success)
```

The `add metadata` command causes rows to be created in the tables `Dublin_Core_Metadata` `Dublin_Core_Metadata_Sub` in the `gwirdsif` database:

```
mysql> select * from gwirdsif.Dublin_Core_Metadata \
   where irods_object_path = '/tempZone/home/lfinsto/abc.txt'\G
⇒
*************************** 1. row ***************************
dublin_core_metadata_id: 1
                user_id: 1
        irods_server_id: 1
      irods_object_path: /tempZone/home/lfinsto/abc.txt
              handle_id: 61
                deleted: 0
                created: 2013-08-30 12:56:45
          last_modified: 0000-00-00 00:00:00
1 row in set (0.00 sec)

mysql> select * from gwirdsif.Dublin_Core_Metadata_Sub \
   where dublin_core_metadata_id=1 \
   order by dublin_core_metadata_sub_id\G
⇒
*************************** 1. row ***************************
dublin_core_metadata_sub_id: 1
    dublin_core_metadata_id: 1
     dublin_core_element_id: 1
        dublin_core_term_id: 0
                      value: Sample Dublin Core Metadata (Title)
*************************** 2. row ***************************
dublin_core_metadata_sub_id: 2
    dublin_core_metadata_id: 1
     dublin_core_element_id: 2
        dublin_core_term_id: 0
                      value: Laurence D. Finston (Creator)
[...]
16 rows in set (0.00 sec)
```

The following MySQL query prints out the data in a more informative way, using fields from the tables `Users`, `Dublin_Core_Elements` and `Dublin_Core_Terms`. (Redundant information has been replaced by "[. . .]" after the first row.)

```
select U.user_id, U.username, M.dublin_core_metadata_id,
    M.irods_server_id, M.irods_object_path, M.handle_id,
    S.dublin_core_metadata_sub_id,
    S.dublin_core_element_id, E.element_name,
```

```
    S.dublin_core_term_id, T.term_name,
    S.value
    from Users as U, Dublin_Core_Metadata as M,
    Dublin_Core_Metadata_Sub as S,
    Dublin_Core_Elements as E, Dublin_Core_Terms as T
    where U.user_id = M.user_id
    and M.dublin_core_metadata_id = 1
    and M.dublin_core_metadata_id = S.dublin_core_metadata_id
    and S.dublin_core_element_id = E.dublin_core_element_id
    and S.dublin_core_term_id = T.dublin_core_term_id
    order by S.dublin_core_metadata_sub_id, S.dublin_core_element_id, \
       S.dublin_core_term_id\G
⇒
************ 1. row ***************************
                   user_id: 1
                  username: lfinsto
    dublin_core_metadata_id: 1
            irods_server_id: 1
          irods_object_path: /tempZone/home/lfinsto/abc.txt
                  handle_id: 61
dublin_core_metadata_sub_id: 1
     dublin_core_element_id: 1
               element_name: title
        dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Sample Dublin Core Metadata (Title)
*************************** 2. row ***************************
[...]
dublin_core_metadata_sub_id: 2
     dublin_core_element_id: 2
               element_name: creator
        dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Laurence D. Finston (Creator)
*************************** 3. row ***************************
[...]
dublin_core_metadata_sub_id: 3
     dublin_core_element_id: 3
               element_name: subject
        dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Sample Dublin Core Metadata 1 (Subject)
```

```
*************************** 4. row ***************************
[...]
dublin_core_metadata_sub_id: 4
     dublin_core_element_id: 4
               element_name: description
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Sample Dublin Core Metadata 1 (Description)
*************************** 5. row ***************************
[...]
dublin_core_metadata_sub_id: 5
     dublin_core_element_id: 5
               element_name: publisher
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: GWDG 1 (Publisher)
*************************** 6. row ***************************
[...]
dublin_core_metadata_sub_id: 6
     dublin_core_element_id: 6
               element_name: contributor
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Sample contributor 1
*************************** 7. row ***************************
[...]
dublin_core_metadata_sub_id: 7
     dublin_core_element_id: 7
               element_name: date
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: 2012-12-06 12:11:26
*************************** 8. row ***************************
[...]
dublin_core_metadata_sub_id: 8
     dublin_core_element_id: 8
               element_name: type
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: iRODS object (Type)
```

```
*************************** 9. row ***************************
[...]
dublin_core_metadata_sub_id: 9
     dublin_core_element_id: 9
               element_name: format
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: ASCII (Format)
*************************** 10. row ***************************
[...]
dublin_core_metadata_sub_id: 10
     dublin_core_element_id: 10
               element_name: identifier
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: XXX (Identifier)
*************************** 11. row ***************************
[...]
dublin_core_metadata_sub_id: 11
     dublin_core_element_id: 11
               element_name: source
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: GWDG (Source)
*************************** 12. row ***************************
[...]
dublin_core_metadata_sub_id: 12
     dublin_core_element_id: 12
               element_name: language
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: English (Language)
*************************** 13. row ***************************
[...]
dublin_core_metadata_sub_id: 13
     dublin_core_element_id: 13
               element_name: relation
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Not applicable (Relation)
```

```
*************************** 14. row ***************************
[...]
dublin_core_metadata_sub_id: 14
     dublin_core_element_id: 14
               element_name: coverage
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: Not applicable (Coverage)
*************************** 15. row ***************************
[...]
dublin_core_metadata_sub_id: 15
     dublin_core_element_id: 15
               element_name: rights
          dublin_core_term_id: 0
                  term_name: NULL_DUBLIN_CORE_TERM
                      value: All rights reserved (Rights)
*************************** 16. row ***************************
[...]
dublin_core_metadata_sub_id: 16
     dublin_core_element_id: 0
               element_name: NULL_DUBLIN_CORE_ELEMENT
          dublin_core_term_id: 1
                  term_name: abstract
                      value: Sample Abstract
16 rows in set (0.01 sec)
```

As indicated in the client-side terminal output from the `add metadata` command, a handle, '`12345/00002`', is added for the Dublin Core (DC) metadata and an AVU is created for the iRODS object '`abc.txt`':

```
get handle pid 12345/00002
⇒
get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            11
type:                           IRODS_OBJECT_PID
data_length:                    11
data:                           12345/00001
[...]
timestamp:                      1377860205 (2013-08-30 12:56:45 CEST)
[...]
handle_id:                      61
handle_value_id:                141
irods_object_id:                0
created:                        1377860205 (2013-08-30 12:56:45 CEST)
last_modified:                  0
```

```
delete_from_database_timestamp:  0
created_by_user:                 1
marked_for_deletion:             0

get handle -->
Response code:                   0
filename:
handle:                          12345/00002
idx:                             21
type:                            IRODS_OBJECT_REF
data_length:                     30
data:                            /tempZone/home/lfinsto/abc.txt
[...]
handle_id:                       61
[...]

get handle -->
Response code:                   0
filename:
handle:                          12345/00002
idx:                             91
type:                            DC_METADATA
data_length:                     83
data:                            Qualified Dublin Core XML Metadata \
   for iRODS object /tempZone/home/lfinsto/abc.txt.
[...]
handle_value_id:                 140
[...]

get handle -->
Response code:                   0
filename:
handle:                          12345/00002
idx:                             300
type:                            HS_ADMIN
data_length:                     22
data:                            ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
handle_value_id:                 139
[...]
```

This handle contains handle values referring both to the iRODS object '`/tempZone/home/lfinsto/abc.txt`' and the handle for the latter, namely '`12345/00001`'.

```
get metadata abc.txt
⇒
```

```
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  2

Irods_Object_Type:
id ==                                  0
path ==                                /tempZone/home/lfinsto/abc.txt
[...]
avu_vector.size() == 2
Showing avu_vector:
Irods_AVU_Type:
id ==                        0
irods_object_id ==           0
user_id ==                   0
irods_object_path ==         (empty)
attribute ==                 PID
value ==                     12345/00001
units ==                     (empty)
time_set ==                  1367603247 (seconds since epoch):  \
   2013-05-03 19:47:27 CEST +0200
deleted_from_archive ==      0
deleted_from_gwirdsif_db == 0

Irods_AVU_Type:
id ==                        0
irods_object_id ==           0
user_id ==                   0
irods_object_path ==         (empty)
attribute ==                 DC_METADATA_PID
value ==                     12345/00002
units ==                     (empty)
time_set ==                  1369297602 (seconds since epoch):  \
   2013-05-23 10:26:42 CEST +0200
deleted_from_archive ==      0
deleted_from_gwirdsif_db == 0


[...]

Received metadata for iRODS object '/tempZone/home/lfinsto/abc.txt'.
Stored in temporary file:  /tmp/gwirdcli.00K8FW
```

The `get metadata` command tells the server to send any Dublin Core metadata for the iRODS object to the client, which stores it in a temporary file, here, '`/tmp/gwirdcli.00K8FW`':

```
cat /tmp/gwirdcli.00K8FW
```

```
⇒
<?xml version="1.0"?>

<metadata
  xmlns="http://example.org/myapp/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/myapp/ \
     http://example.org/myapp/schema.xsd"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/">

  <dc:title xsi:type="title attribute" xsi:typex="title attribute 2">
    Sample Dublin Core Metadata (Title)
  </dc:title>
  <dc:creator>
    Laurence D. Finston (Creator)
  </dc:creator>
[...]
```

Except for a couple of blank lines at the end of '`metadata_sample_1.xml`', the two files are identical:

```
diff --brief /tmp/gwirdcli.00K8FW metadata_sample_1.xml
⇒
Files /tmp/gwirdcli.00K8FW and metadata_sample_1.xml differ

diff --ignore-blank-lines --brief \
   /tmp/gwirdcli.00K8FW metadata_sample_1.xml; echo $?
⇒
0
```

When the server converted the contents of '`metadata_sample_1.xml`' to database entries, it ignored the trailing blank lines. When it reversed the procedure to generate a text to send to the client, nothing more was known on the server-side about the blank lines in the original file (nor was there any reason for there to be).

If desired, the Dublin Core metadata can additionally be stored in an iRODS object of its own. To do this, call `add metadata` with the '`store`' option:

```
add metadata metadata_sample_1.xml abc.txt store
⇒
add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle for metadata:  12345/00002.

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
```

```
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00001’ with type \
   ‘IRODS_OBJECT_PID’ to handle ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for iRODS object \
      ‘/tempZone/home/lfinsto/abc.txt’ with type ‘IRODS_OBJECT_REF’ \
      to handle ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00002’ with type \
      ‘DC_METADATA_PID’ to handle ‘12345/00001’ successfully

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Call to ‘imeta’ succeeded.  Added AVU with type ‘DC_METADATA_PID’ \
      and value ‘12345/00002’ to iRODS object \
      ‘/tempZone/home/lfinsto/abc.txt’.

store metadata-->
Exit status:                              0
Dublin Core metadata/iRODS object file:   \
   /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object referred to:                 /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle 12345/00003 for Dublin Core metadata iRODS object \
      ‘/tempZone/home/lfinsto/metadata_sample_1.xml’.

add metadata-->
Exit status:                              0
Metadata file                             metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Stored Dublin Core metadata in iRODS object \
      ‘/tempZone/home/lfinsto/metadata_sample_1.xml’ successfully.
```

```
add metadata-->
Exit status:                                0
Metadata file                               metadata_sample_1.xml
iRODS object                                /tempZone/home/lfinsto/abc.txt
Server message:
   (Success)
```

The `ls` command shows that the server has created an iRODS object '`metadata_sample_1.xml`' for me in my current working iRODS directory:

```
echo "ls" | gwirdcli --suppress-prompt localhost
⇒
ls -->
/tempZone/home/lfinsto:
  abc.txt
  metadata_sample_1.xml
```

Yet another handle, '`12345/00003`', has been created for this iRODS object, in addition to '`12345/00002`', which refers to the Dublin Core metadata stored in the `gwirdsif.Dublin_Core_Metadata` and `gwirdsif.Dublin_Core_Metadata_Sub` database tables:

```
get handle pid 12345/00003
⇒
get handle -->
Response code:                  0
filename:
handle:                         12345/00003
idx:                            11
type:                           IRODS_OBJECT_PID
data_length:                    11
data:                           12345/00001
[...]
timestamp:                      1377863624 (2013-08-30 13:53:44 CEST)
[...]
handle_id:                      62
handle_value_id:                147
irods_object_id:                0
created:                        1377863624 (2013-08-30 13:53:44 CEST)
last_modified:                  0
delete_from_database_timestamp: 0
created_by_user:                1
marked_for_deletion:            0

get handle -->
Response code:                  0
filename:
handle:                         12345/00003
idx:                            21
```

```
type:                           IRODS_OBJECT_REF
data_length:                    30
data:                           /tempZone/home/lfinsto/abc.txt
[...]
handle_value_id:                146
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00003
idx:                            101
type:                           DC_METADATA_PID
data_length:                    11
data:                           12345/00002
[...]
handle_value_id:                148
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00003
idx:                            121
type:                           DC_METADATA_IRODS_OBJECT
data_length:                    44
data: \
   /tempZone/home/lfinsto/metadata_sample_1.xml
[...]
handle_value_id:                145
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00003
idx:                            300
type:                           HS_ADMIN
data_length:                    22
data:                           ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
handle_value_id:                144
[...]
```

This handle contains a handle value with index (`idx`) 121, type `DC_METADATA_IRODS_OBJECT` and data '`/tempZone/home/lfinsto/metadata_sample_1.xml`', as well as handle values re-

ferring to the iRODS object '`/tempZone/home/lfinsto/abc.txt`', its handle '`12345/00001`' and finally the handle '`12345/00002`' for the Dublin Core metadata stored in the `gwirdsif` database.

AVUs have been created for the iRODS object '`/tempZone/home/lfinsto/metadata_sample_1.xml`' referring to it's own handle '`12345/00003`', the handle '`12345/00002`' for the Dublin Core metadata in the `gwirdsif` database, the iRODS object '`/tempZone/home/lfinsto/abc.txt`', and the latter's handle '`12345/00001`':

```
get metadata metadata_sample_1.xml
⇒
get metadata -->
Filename:        /tempZone/home/lfinsto/metadata_sample_1.xml
Exit status:     0
Number of AVUs:  5

Irods_Object_Type:
id ==                                    0
path == \
   /tempZone/home/lfinsto/metadata_sample_1.xml
[...]
avu_vector.size() == 5
Showing avu_vector:
Irods_AVU_Type:
id ==                        0
irods_object_id ==           0
user_id ==                   0
irods_object_path ==         (empty)
attribute ==                 IRODS_OBJECT_PID
value ==                     12345/00001
units ==                     (empty)
time_set ==                  1369397378 (seconds since epoch): \
   2013-05-24 14:09:38 CEST +0200
deleted_from_archive ==      0
deleted_from_gwirdsif_db == 0

Irods_AVU_Type:
id ==                        0
irods_object_id ==           0
user_id ==                   0
irods_object_path ==         (empty)
attribute ==                 IRODS_OBJECT_REF
value ==                     /tempZone/home/lfinsto/abc.txt
[...]

Irods_AVU_Type:
id ==                        0
```

```
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_PID
value ==                    12345/00002
[...]

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                PID
value ==                    12345/00003
[...]

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                TYPE
value ==                    DC_METADATA_IRODS_OBJECT
[...]

[...]
```

Handle values referring to the *Dublin Core metadata iRODS object* '`/tempZone/home/lfinsto/metadata_sample_1.xml`' and its handle '`12345/00003`' have also been added to the handles '`12345/00001`' and '`12345/00002`':

```
get handle pid 12345/00001
get handle pid 12345/00002
⇒
get handle -->
Response code:                  0
filename:                       /tempZone/home/lfinsto/abc.txt
handle:                         12345/00001
idx:                            1
type:                           IRODS_OBJECT
data_length:                    30
data:                           /tempZone/home/lfinsto/abc.txt
[...]
timestamp:                      1377863607 (2013-08-30 13:53:27 CEST)
[...]
handle_id:                      60
handle_value_id:                137
irods_object_id:                0
```

```
created:                        1377863607 (2013-08-30 13:53:27 CEST)
last_modified:                  0
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            101
type:                           DC_METADATA_PID
data_length:                    11
data:                           12345/00002
[...]
handle_value_id:                143
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            131
type:                           DC_METADATA_IRODS_OBJECT_PID
data_length:                    11
data:                           12345/00003
[...]
handle_value_id:                150
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            141
type:                           DC_METADATA_IRODS_OBJECT_REF
data_length:                    44
data: \
   /tempZone/home/lfinsto/metadata_sample_1.xml
[...]
handle_value_id:                149
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            211
type:                           CREATOR
```

```
data_length:                    68
data: \
   /C=DE/O=GWDG/OU=gwrdifpk/L=Goettingen/ST=Germany/CN=Laurence Finston
[...]
handle_value_id:                138
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            300
type:                           HS_ADMIN
data_length:                    22
data:                           ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
handle_value_id:                136
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            11
type:                           IRODS_OBJECT_PID
data_length:                    11
data:                           12345/00001
[...]
handle_value_id:                141
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            21
type:                           IRODS_OBJECT_REF
data_length:                    30
data:                           /tempZone/home/lfinsto/abc.txt
[...]
handle_value_id:                142
[...]

get handle -->
Response code:                  0
filename:
```

```
handle:                         12345/00002
idx:                            91
type:                           DC_METADATA
data_length:                    83
data: \
   Qualified Dublin Core XML Metadata for iRODS object \
   /tempZone/home/lfinsto/abc.txt.
[...]
handle_value_id:                140
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            131
type:                           DC_METADATA_IRODS_OBJECT_PID
data_length:                    11
data:                           12345/00003
[...]
handle_value_id:                152
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            141
type:                           DC_METADATA_IRODS_OBJECT_REF
data_length:                    44
data: \
   /tempZone/home/lfinsto/metadata_sample_1.xml
[...]
handle_value_id:                151
[...]

get handle -->
Response code:                  0
filename:
handle:                         12345/00002
idx:                            300
type:                           HS_ADMIN
data_length:                    22
data:                           ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
handle_value_id:                139
```

```
[...]
```

AVUs have also been added to the iRODS object '/tempZone/home/lfinsto/abc.txt':

```
get metadata abc.txt
⇒
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  4

Irods_Object_Type:
id ==                                   0
path ==                                 /tempZone/home/lfinsto/abc.txt
[...]
avu_vector.size() == 4
Showing avu_vector:
Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_IRODS_OBJECT_REF
value ==                    /tempZone/home/lfinsto/metadata_sample_1.xml
units ==                    (empty)
time_set ==                 1369400978 (seconds since epoch): \
   2013-05-24 15:09:38 CEST +0200
deleted_from_archive ==     0
deleted_from_gwirdsif_db == 0

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_IRODS_OBJECT_PID
value ==                    12345/00003
[...]

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_PID
value ==                    12345/00002
[...]
```

```
Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                PID
value ==                    12345/00001
[...]

[...]

Received metadata for iRODS object '/tempZone/home/lfinsto/abc.txt'.
Stored in temporary file:  /tmp/gwirdcli.1AX3MU
```

Finally, AVUs have been added to the iRODS object '`/tempZone/home/lfinsto/abc.txt`' referring to '`metadata_sample_1.xml`' and its handle '`12345/00003`', as well as to the handle for the Dublin Core metadata in the `gwirdsif` database, namely '`12345/00002`':

```
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  4

Irods_Object_Type:
id ==                                   0
path ==                                 /tempZone/home/lfinsto/abc.txt
[...]
avu_vector.size() == 4
Showing avu_vector:
Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_IRODS_OBJECT_REF
value ==                    /tempZone/home/lfinsto/metadata_sample_1.xml
units ==                    (empty)
time_set ==                 1369397378 (seconds since epoch): \
   2013-05-24 14:09:38 CEST +0200
deleted_from_archive ==     0
deleted_from_gwirdsif_db == 0

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_IRODS_OBJECT_PID
```

```
value ==                    12345/00003
[...]

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                DC_METADATA_PID
value ==                    12345/00002
[...]

Irods_AVU_Type:
id ==                       0
irods_object_id ==          0
user_id ==                  0
irods_object_path ==        (empty)
attribute ==                PID
value ==                    12345/00001
[...]

[...]
```

4.2.4 Deleting and Undeleting

In most of the following examples, and in most of the ones in subsequent chapters, only the commands are shown, not the calls to `gwirdcli`, as in the examples above. For information on how to invoke `gwirdcli`, see Section 4.2.1 [Invoking gwirdcli], page 16, above.

4.2.4.1 iRODS Objects

If I decide that I no longer need the remote iRODS object '`abc.txt`', I can delete it with the `rm` command:

```
rm abc.txt
⇒
Mark iRODS object for deletion response -->
iRODS object path(s):          /tempZone/home/lfinsto/abc.txt
Response code:                 0
Options:                       0
Marked for deletion from archive.
Message:                       Success
```

Huh? It was just "marked for deletion"?! I can use the `ls` command to check if it still exists:

```
echo "ls abc.txt" | gwirdcli --suppress-prompt localhost
⇒
ls -->
ERROR: lsUtil: srcPath /tempZone/home/lfinsto/abc.txt does not exist \
   or user lacks access permission
```

No, it was deleted alright.

When the `rm` command is used without any options, the iRODS object is "marked for immediate deletion". If `gwirdsif` has been invoked in such a way that *purging* the iRODS archive is enabled, then a thread in which a function for this purpose is running will be "woken up" and the iRODS object will be deleted.

Once an iRODS object has been deleted, however, it cannot be recovered, unless an external backup system is used. It may therefore often be useful to delay deletion for a period of time, in order to give oneself time to reconsider. During this "window" of time, the iRODS object can be "undeleted" using the `undelete` command.

In order to be able to do this, `rm` must be called using the '`delay`' option, which takes an optional argument:

```
echo "rm --delay abc.txt" | gwirdcli --suppress-prompt localhost
⇒
Mark iRODS object for deletion response -->
iRODS object path(s):          /tempZone/home/lfinsto/abc.txt
Response code:                 0
Options:                       1
Marked for deletion from archive.
Timestamp (deletion time):     1377789922 == 2013-08-29 17:25:22 CEST
Message:                       Success
```

The "deletion time" is 2013-08-29 17:25:22 CEST, which happens to be the current time when the author typed this paragraph. However, the time that the iRODS object will actually be deleted depends on the server-side parameter *purge_irods_archive_limit*. This parameter can be set by using the command-line option '`--purge-irods-archive-limit`' when starting the server. Its default value is 172,800, which is two days in seconds. The function that actually deletes the iRODS objects won't do so until *purge_irods_archive_limit* seconds have elapsed since the time stored in the "deletion time" timestamp.

Users of the client, however, will not necessarily know the value of *purge_irods_archive_limit*, which may also differ from run-to-run of the server. In addition, there is no way at present for a user to query the server for the current value of *purge_irods_archive_limit*. If I want to be sure that an iRODS object won't be deleted for, say, a week, then I can call the `rm` command with the '`--delay`' and an argument to the latter:

```
echo "rm --delay 7: abc.txt" | gwirdcli --suppress-prompt localhost
⇒
Mark iRODS object for deletion response -->
iRODS object path(s):          /tempZone/home/lfinsto/abc.txt
Response code:                 0
Options:                       1
Marked for deletion from archive.
Timestamp (deletion time):     1378395406 == 2013-09-05 17:36:46 CEST
Message:                       Success
```

The argument '`7:`' tells the server to set the deletion time timestamp to a time seven days in the future. Now, the purge function will delete the iRODS object at the soonest *purge_irods_archive_limit* seconds after this time.

Please note that if *purge_irods_archive_limit* is set to 0, then all iRODS objects that have been marked for deletion and whose timestamps are earlier than the current time will be deleted as soon as the purge function runs. This will be all iRODS objects that have been deleted without specifying a delay plus those whose "delay" has expired.

The way iRODS objects are purged is also influenced by a second parameter, namely *purge_irods_archive_interval*, which can be set using the command-line option '`--purge-irods-archive-interval`'. The function `purge_irods_archive` runs in an endless loop. *purge_irods_archive_interval* is the time in seconds that the purge thread "sleeps" between iterations of this loop, unless it's "woken up" for an immediate deletion. The default value of *purge_irods_archive_interval* is 3600, i.e., one hour in seconds. Purging can be suppressed entirely by invoking `gwirdsif` with 0 as the argument to the '`--purge-irods-archive-interval`' option:

```
gwirdsif --purge-irods-archive-interval 0 [...]
```

In this case, the purge thread won't be started at all.

One consequence of performing the actual deletion in a separate thread is that the thread function has no connection with any *session* in which the server is communicating with the client. That is, the purge thread has no way of communicating with the owner of the iRODS objects, who may not even be communicating with the server when the actual deletion takes place. It is therefore not possible for the server to send a response to the client informing the latter when deletion has taken place.

If I now decide that deleting '`abc.txt`' was a mistake, after all, I can cancel the pending deletion with the `undelete` command:

```
echo "undelete abc.txt" | gwirdcli --suppress-prompt localhost
⇒
Undelete response -->
Response code:         0
iRODS object name(s):  /tempZone/home/lfinsto/abc.txt
Message:               Undeleted iRODS objects successfully
mysql> select * from gwirdsif.Irods_Objects where irods_object_id > 0\G
⇒
*************************** 1. row ***************************
                    irods_object_id: 1
                            user_id: 1
                    irods_server_id: 1
                   irods_object_path: /tempZone/home/lfinsto/abc.txt
    marked_for_deletion_from_archive: 0
                deleted_from_archive: 0
       delete_from_archive_timestamp: 0000-00-00 00:00:00
marked_for_deletion_from_gwirdsif_db: 0
   delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
                             created: 2013-08-29 17:36:45
                       last_modified: 2013-08-29 18:05:37
1 row in set (0.00 sec)
```

In the row for '`abc.txt`' in the '`gwirdsif.Irods_Objects`' database table, the fields '`marked_for_deletion_from_gwirdsif_db`' and '`delete_from_archive_timestamp`' have

been reset to 0 (the latter displayed as a timestamp), and the field '`last_modified`' has been updated. Now, when the purging function next runs, it will not delete '`abc.txt`'.

4.2.4.2 Handles

Handles, too, can be deleted and undeleted:

```
delete handle "12345/00001"
⇒
Delete handle response -->
Response code:  0 (Success)
Handle:         12345/00001
Message:        Marked 4 rows for deletion from \
   'handlesystem_standalone.handles' database table
```

As with iRODS objects, handles are first marked for deletion and then finally deleted by a "purging" function in a separate thread, whereby different threads and functions are used for iRODS objects and handles.

Analogous to the way purging is managed for iRODS objects (see Section 4.2.4.1 [iRODS Objects], page 46), the way handles are purged is controlled by two parameters, namely *purge_database_interval* and *purge_database_limit*, which can be set using the command-line options '`--purge-database-interval`' and '`--purge-database-limit`', respectively.

Purging handles is simpler than purging iRODS objects because the former exist only in the form of rows in a database table, whereas the database entries for iRODS objects refer to objects within the iRODS system, i.e., external to the `gwrdifpk` system. The handle database may either be '`handlesystem`' for handles with prefixes registered with CNRI's Handle System, or '`handlesystem_standalone`' for a *standalone* handle service using private prefixes and not integrated with the global Handle System.

```
get handle pid 12345/00001
⇒
get handle -->
Response code:                  0
filename:                       /tempZone/home/lfinsto/abc.txt
handle:                         12345/00001
idx:                            1
type:                           IRODS_OBJECT
data_length:                    30
data:                           /tempZone/home/lfinsto/abc.txt
[...]
timestamp:                      1377790605 (2013-08-29 17:36:45 CEST)
[...]
handle_id:                      56
handle_value_id:                130
irods_object_id:                0
created:                        1377790605 (2013-08-29 17:36:45 CEST)
last_modified:                  1377850629 (2013-08-30 10:17:09 CEST)
delete_from_database_timestamp: 1377850629 (2013-08-30 10:17:09 CEST)
created_by_user:                1
marked_for_deletion:            1
```

```
get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            211
type:                           CREATOR
data_length:                    68
data: \
   /C=DE/O=GWDG/OU=gwrdifpk/L=Goettingen/ST=Germany/CN=Laurence Finston
[...]
timestamp:                      1377790605 (2013-08-29 17:36:45 CEST)
[...]
handle_id:                      56
handle_value_id:                131
irods_object_id:                0
created:                        1377790605 (2013-08-29 17:36:45 CEST)
last_modified:                  1377850629 (2013-08-30 10:17:09 CEST)
delete_from_database_timestamp: 1377850629 (2013-08-30 10:17:09 CEST)
created_by_user:                1
marked_for_deletion:            1

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            231
type:                           HANDLE_MARKED_FOR_DELETION
data_length:                    23
data:                           2013-08-30 08:17:09 UTC
[...]
timestamp:                      1377850629 (2013-08-30 10:17:09 CEST)
[...]
handle_id:                      56
handle_value_id:                132
irods_object_id:                0
created:                        1377850629 (2013-08-30 10:17:09 CEST)
last_modified:                  1377850629 (2013-08-30 10:17:09 CEST)
delete_from_database_timestamp: 1377850629 (2013-08-30 10:17:09 CEST)
created_by_user:                1
marked_for_deletion:            1

get handle -->
Response code:                  0
filename:
handle:                         12345/00001
idx:                            300
```

```
type:                              HS_ADMIN
data_length:                       22
data:                              ^G\363^@^@^@
0.NA/12345^@^@^@\310^@^@
[...]
timestamp:                         1377790605 (2013-08-29 17:36:45 CEST)
[...]
handle_id:                         56
handle_value_id:                   129
irods_object_id:                   0
created:                           1377790605 (2013-08-29 17:36:45 CEST)
last_modified:                     1377850629 (2013-08-30 10:17:09 CEST)
delete_from_database_timestamp:    1377850629 (2013-08-30 10:17:09 CEST)
created_by_user:                   1
marked_for_deletion:               1
```

In all of the handle values, the field '`marked_for_deletion`' has been set to 1 and the fields '`delete_from_database_timestamp`' and '`last_modified`' have been set to the current time. Furthermore, a handle value with index ('`idx`') 231 and type `HANDLE_MARKED_FOR_DELETION` has been added to the handle.

If purging the database hasn't been disabled by calling `gwirdsif` with '`--purge-database-interval 0`', then the thread function `purge_server_database` will delete the handle when it next runs after the limit *purge_database_limit* has expired.

Let's say the limit has expired and `purge_server_database` has deleted the handle. Now, trying to retrieve the handle fails:

```
get handle pid 12345/00001
⇒
get handle -->
Server-side error:
Response code:   3
handle:          12345/00001
Database query returned 0 rows
```

Remember the AVU that was created for '`abc.txt`' to store the PID? It's been deleted, too:

```
get metadata abc.txt
⇒
get metadata -->
Filename:        /tempZone/home/lfinsto/abc.txt
Exit status:     0
Number of AVUs:  0

No user-defined metadata (AVUs) to display
```

Please note that whereas iRODS objects are marked for immediate deletion by default and any delay must be specified with the '`delay`' option, handles are marked for delayed deletion by default. If immediate deletion is desired for a handle, then this must be specified using the '`immediate`' option.

The reason for this difference is that the `gwrdifpk` commands based on iRODS' icommands, such as `rm`, based on `irm`, are intended to function as much like their models as possible and not to diverge unless there's a good reason. `irm` deletes an iRODS object immediately; there is no concept of delayed deletion in iRODS itself, at least on the level of the icommands interface.

The author is considering implementing a `delete` command for iRODS objects, analogous to `delete` for handles, i.e., with delayed deletion by default.

4.2.4.3 Dublin Core metadata

Dublin Core metadata can also be deleted, whereby the procedure is similar to that used for iRODS objects and handles. An added complication is that Dublin Core metadata may also be stored in an iRODS object of its own. When deleting, users may specify whether the database entry containing the metadata is deleted, or the iRODS object, or both.

Example:

```
put -f +pid +gen abc.txt
add metadata "metadata_sample_1.xml" "abc.txt"

(Client output:)

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     'iput' command succeeded, returning 0

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Success:  Generated PID '12345/00001'

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Added handle values with type == 'IRODS_OBJECT' \
                 and type == 'CREATOR_INDEX' successfully

put -->
Filename:     /tempZone/home/lfinsto/abc.txt
Exit status:  0
Response:     Success:  Stored PID '12345/00001' in iRODS object metadata

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle for metadata:  12345/00002.

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle '12345/00001' with type 'IRODS_OBJECT_PID' \
      to handle '12345/00002' successfully
```

```
add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for iRODS object '/tempZone/home/lfinsto/abc.txt' \
      with type 'IRODS_OBJECT_REF' to handle '12345/00002' successfully

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle '12345/00002' with type 'DC_METADATA_PID' \
      to handle '12345/00001' successfully

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Call to 'imeta' succeeded.  Added AVU with type 'DC_METADATA_PID' \
      and value '12345/00002' to iRODS object '/tempZone/home/lfinsto/abc.txt'.

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   (Success)
```

An iRODS object '`abc.txt`', the handles '`12345/00001`' and '`12345/00001`' and rows have been created in the tables `Dublin_Core_Metadata` and `Dublin_Core_Metadata_Sub` in the `gwirdsif` database:

```
mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G

⇒

*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
                irods_server_id: 1
              irods_object_path: /tempZone/home/lfinsto/abc.txt
                      handle_id: 118
            marked_for_deletion: 0
                        created: 2013-12-04 14:36:45
                  last_modified: 0000-00-00 00:00:00
delete_from_database_timestamp: 0000-00-00 00:00:00
                    delete_file: 0
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
           irods_object_ref_id: 1
          irods_object_self_id: 0
1 row in set (0.00 sec)

mysql> select * from Dublin_Core_Metadata_Sub where dublin_core_metadata_id = 1
       order by dublin_core_metadata_sub_id\G
```

```
⇒

*************************** 1. row ***************************
dublin_core_metadata_sub_id: 1
    dublin_core_metadata_id: 1
     dublin_core_element_id: 1
        dublin_core_term_id: 0
                      value: Sample Dublin Core Metadata (Title)
*************************** 2. row ***************************
dublin_core_metadata_sub_id: 2
    dublin_core_metadata_id: 1
     dublin_core_element_id: 2
        dublin_core_term_id: 0
                      value: Laurence D. Finston (Creator)
*************************** 3. row ***************************
dublin_core_metadata_sub_id: 3
    dublin_core_metadata_id: 1
     dublin_core_element_id: 3
        dublin_core_term_id: 0
                      value: Sample Dublin Core Metadata 1 (Subject)

[...]

16 rows in set (0.00 sec)
```

In the row for the iRODS object '`abc.txt`' in the `Irods_Objects` database table, the field `dublin_core_metadata_id` contains a reference to the `dublin_core_metadata_id` field of the row in the `Dublin_Core_Metadata` table:

```
mysql> select * from Irods_Objects where irods_object_id > 0\G
*************************** 1. row ***************************
                    irods_object_id: 1
                            user_id: 1
                    irods_server_id: 1
                  irods_object_path: /tempZone/home/lfinsto/abc.txt
    marked_for_deletion_from_archive: 0
                deleted_from_archive: 0
       delete_from_archive_timestamp: 0000-00-00 00:00:00
marked_for_deletion_from_gwirdsif_db: 0
   delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
                             created: 2013-12-04 14:36:45
                       last_modified: 0000-00-00 00:00:00
             dublin_core_metadata_id: 1
dublin_core_metadata_irods_object_id: 0
                 irods_object_ref_id: 0
1 row in set (0.00 sec)
```

The command `delete metadata abc.txt` does not cause the Dublin Core metadata to be deleted immediately, but rather *marks it for deletion* by setting the value of the `marked_for_deletion` field to 1 and that of the `delete_from_database_timestamp` and `last_modified` fields to the current time:

```
delete metadata abc.txt
```

(Client output:)

```
Delete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Options:                      0 (00000000)
```

```
Message:                      Success

mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G

⇒

*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
                 irods_server_id: 1
               irods_object_path: /tempZone/home/lfinsto/abc.txt
                       handle_id: 118
             marked_for_deletion: 1
                         created: 2013-12-04 14:36:45
                   last_modified: 2013-12-05 10:20:18
delete_from_database_timestamp: 2013-12-05 10:20:18
                    delete_file: 0
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
            irods_object_ref_id: 1
           irods_object_self_id: 0
1 row in set (0.00 sec)
```

This row will not be deleted until *at least* `purge_dc_metadata_limit` seconds have passed since the time stored in the `delete_from_database_timestamp` field. The global variable `purge_dc_metadata_limit` is set on the server-side by means of the command-line option '`--purge-dc-metadata-limit`'. After `purge_dc_metadata_limit` have passed, the row will be deleted the next time the thread function `purge_dc_metadata` runs. The command-line option '`--purge-metadata-interval`' sets the global variable `purge_dc_metadata_interval`, which controls how often `purge_dc_metadata` runs. In particular, if `gwirdsif` has been invoked with '`--purge-dc-metadata-interval 0`', `purge_dc_metadata` will never run and the row will never be deleted. See Section 13.2 [Global variables], page 102, and Section 6.2.10 [Purging options], page 69.

So long as the row has not "expired", as described above, the user may "undelete" it by means of the `undelete metadata` command:

```
undelete metadata abc.txt
```

(Client output:)

```
Undelete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Message:                      Success

mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G
*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
                 irods_server_id: 1
               irods_object_path: /tempZone/home/lfinsto/abc.txt
                       handle_id: 118
             marked_for_deletion: 0
                         created: 2013-12-04 14:36:45
                   last_modified: 2013-12-05 10:39:29
delete_from_database_timestamp: 0000-00-00 00:00:00
                    delete_file: 0
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
```

```
           irods_object_ref_id: 1
          irods_object_self_id: 0
1 row in set (0.00 sec)
```

The value of the `marked_for_deletion` field has been reset to 0, the `delete_from_database_timestamp` has been reset to '0000-00-00 00:00:00', i.e., the zero timestamp, and the value of the `last_modified` field has been updated to the current date and time.

Assuming that the Dublin Core metadata is *not* "undeleted", the rows in the `Dublin_Core_Metadata` and `Dublin_Core_Metadata_Sub` tables in the `gwirdsif` database will be deleted:

```
mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G
Empty set (0.00 sec)
```

In addition, in the `Irods_Objects` table, the `dublin_core_metadata_id` has been reset to 0:

```
mysql> select * from Irods_Objects where irods_object_id > 0\G
*************************** 1. row ***************************
                    irods_object_id: 1
                            user_id: 1
                    irods_server_id: 1
                  irods_object_path: /tempZone/home/lfinsto/abc.txt
    marked_for_deletion_from_archive: 0
                deleted_from_archive: 0
       delete_from_archive_timestamp: 0000-00-00 00:00:00
marked_for_deletion_from_gwirdsif_db: 0
   delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
                             created: 2013-12-05 11:01:53
                       last_modified: 2013-12-05 11:07:25
             dublin_core_metadata_id: 0
dublin_core_metadata_irods_object_id: 0
                 irods_object_ref_id: 0
1 row in set (0.00 sec)
```

To delete Dublin Core metadata immediately, the `delete metadata` command may be called with the `immediate` option:

```
delete metadata abc.txt immediate
```

(Client output:)

```
Delete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Options:                      4 (00000100)
Message:                      Success
```

In this case, the `marked_for_deletion` field will be set to 1 and the `delete_from_database_timestamp` will be set to a time 366 days in the past:

```
mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G
*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
               irods_server_id: 1
             irods_object_path: /tempZone/home/lfinsto/abc.txt
                     handle_id: 118
           marked_for_deletion: 1
                       created: 2013-12-05 11:16:28
                 last_modified: 2013-12-05 11:16:31
```

```
delete_from_database_timestamp: 2012-12-04 11:16:31
                   delete_file: 0
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
           irods_object_ref_id: 1
          irods_object_self_id: 0
1 row in set (0.00 sec)
```

If purging Dublin Core metadata has not been disabled by invoking the server with '`--purge-dc-metadata-interval 0`', the function `Dublin_Core_Metadata_Type::mark_dc_metadata_for_deletion` will call `pthread_cond_signal` to "wake up" the thread running `purge_dc_metadata` instead of letting it "sleep" until it would normally "wake up" by itself after `purge_dc_metadata_interval` seconds.

The `add metadata` command may be called with the `store` option:

```
add metadata metadata_sample_1.xml abc.txt store
```

(Client output:)

```
add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle for metadata:  12345/00002.

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00001’ with type ‘IRODS_OBJECT_PID’ to handle \
      ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for iRODS object ‘/tempZone/home/lfinsto/abc.txt’ with type \
      ‘IRODS_OBJECT_REF’ to handle ‘12345/00002’ successfully

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Added handle value for handle ‘12345/00002’ with type ‘DC_METADATA_PID’ to handle \
      ‘12345/00001’ successfully

add metadata-->
Exit status:                              0
Metadata file                             /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                              /tempZone/home/lfinsto/abc.txt
Server message:
   Call to ‘imeta’ succeeded.  Added AVU with type ‘DC_METADATA_PID’ and value \
      ‘12345/00002’ to iRODS object ‘/tempZone/home/lfinsto/abc.txt’.

store metadata-->
```

```
Exit status:                                0
Dublin Core metadata/iRODS object file:     /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object referred to:                   /tempZone/home/lfinsto/abc.txt
Server message:
   Generated handle 12345/00003 for Dublin Core metadata iRODS object \
      '/tempZone/home/lfinsto/metadata_sample_1.xml'.

add metadata-->
Exit status:                                0
Metadata file                               /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                                /tempZone/home/lfinsto/abc.txt
Server message:
   Stored Dublin Core metadata in iRODS object \
      '/tempZone/home/lfinsto/metadata_sample_1.xml' successfully.

add metadata-->
Exit status:                                0
Metadata file                               /tempZone/home/lfinsto/metadata_sample_1.xml
iRODS object                                /tempZone/home/lfinsto/abc.txt
Server message:
   (Success)
```

In this case, the Dublin Core metadata is not only stored in the `gwirdsif` database, but also in an iRODS object, in this example, '`/tempZone/home/lfinsto/metadata_sample_1.xml`'. Handles and AVUs are also created accordingly. In particular, a row is created in the `Irods_Objects` table:

```
mysql> select * from Irods_Objects where irods_object_id > 1\G
*************************** 1. row ***************************
                    irods_object_id: 2
                            user_id: 1
                    irods_server_id: 1
                  irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
    marked_for_deletion_from_archive: 0
                deleted_from_archive: 0
       delete_from_archive_timestamp: 0000-00-00 00:00:00
marked_for_deletion_from_gwirdsif_db: 0
   delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
                             created: 2013-12-05 11:25:02
                       last_modified: 0000-00-00 00:00:00
             dublin_core_metadata_id: 1
dublin_core_metadata_irods_object_id: 0
                 irods_object_ref_id: 1
1 row in set (0.00 sec)
```

The `dublin_core_metadata_id` field contains a reference to the row in the `Dublin_Core_Metadata` table, while the `irods_object_ref_id` field contains one to the row in `Irods_Objects` for '`/tempZone/home/lfinsto/abc.txt`'.

By default, the `delete metadata` command does not delete an iRODS object containing the Dublin Core metadata. To do this, the `file` option must be used:

```
delete metadata abc.txt file
```

(Client output:)

```
Delete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Options:                      1 (00000001)
```

```
Message:                      Success
mysql> select * from Dublin_Core_Metadata where dublin_core_metadata_id = 1\G
*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
                irods_server_id: 1
              irods_object_path: /tempZone/home/lfinsto/abc.txt
                       handle_id: 118
            marked_for_deletion: 1
                        created: 2013-12-05 11:25:02
                  last_modified: 2013-12-05 12:36:17
delete_from_database_timestamp: 2013-12-05 12:36:17
                    delete_file: 1
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
             irods_object_ref_id: 1
            irods_object_self_id: 2
1 row in set (0.00 sec)
```

The `marked_for_deletion` field is set to 1, `delete_from_database_timestamp` and `last_modified` are set to the current time and `delete_file` is set to 1. When `purge_dc_metadata` is run, it will delete the row in `Dublin_Core_Metadata` and the corresponding rows in `Dublin_Core_Metadata_Sub`. In order to delete the iRODS object, it marks the row in the `Irods_Objects` table for immediate deletion and "wakes up" the thread running `purge_irods_archive`, assuming that purging the iRODS archive has not been disabled. See Section 4.2.4.1 [iRODS Objects], page 46.

If the `delete metadata` command is called with the `file_only` option, then only the iRODS object will be marked for deletion, while the rows in the `Dublin_Core_Metadata` and `Dublin_Core_Metadata_Sub` tables will not be:

```
delete metadata abc.txt file_only

Delete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Options:                      2 (00000010)
Message:                      Success
*************************** 1. row ***************************
       dublin_core_metadata_id: 1
                       user_id: 1
                irods_server_id: 1
              irods_object_path: /tempZone/home/lfinsto/abc.txt
                       handle_id: 118
            marked_for_deletion: 1
                        created: 2013-12-05 11:25:02
                  last_modified: 2013-12-05 12:33:25
delete_from_database_timestamp: 2013-12-05 12:33:25
                    delete_file: 2
 dc_metadata_irods_object_path: /tempZone/home/lfinsto/metadata_sample_1.xml
             irods_object_ref_id: 1
            irods_object_self_id: 2
1 row in set (0.00 sec)
```

Here, the `delete_file` field is set to 2.

For more information about the the `delete metadata` command and its options, see Section 8.6 [Dublin Core metadata], page 84. For more information about the function `purge_dc_metadata`, see Section 26.10 [Deleting and rotating files], page 154.

4.3 Concluding Remarks

This chapter has attempted to give an overview of the most important user commands for the client-server application `gwrdifpk`. For this purpose, it was necessary to leave out many details. For further information, please see the following chapters, in particular Chapter 6 [Invoking gwirdsif/gwirdcli], page 63, and Chapter 8 [User commands], page 73.

5 Security considerations

`gwrdifpk` is a package for *long-term* archiving. This implies that security is an important criterion; a company or institution offering this service must be able to guarantee that archived data will remain retrievable and uncorrupted for years: 10, 20, 50 or even longer. Therefore, every effort must be made to ensure both the correct operation of all components belonging to the package and to prevent as far as possible malicious tampering with the data, the components of package itself, or the systems on which it runs.

Unless a private network is available, which is unlikely to be the case, communication between the server and the clients will take place via the internet. While many internet applications operate on the basis of web servers, such as the Apache products HTTPD or Tomcat, this has the disadvantage that such webservers may make the application vulnerable to attack, even if the application itself is programmed in a secure manner (as far as this is possible): If the web server itself is compromised, it is likely that any applications that run under its control will be compromised as well. This will certainly be true if an attacker obtains *root privileges* on the machine where the webserver is running.

Thus, `gwrdifpk` is *not* a web application and does not depend on a webserver. In production use, the server program accepts contacts from clients via a single designated port (5557 by default). Ideally, this would be the only port open to the "outside world" on the machine where the server program runs. Then, the `gwirdsif` process would be the only one that could be attacked from outside. Since all input is read piecewise into fixed-size buffers and excessive input should always cause an error, buffer overflows at least should not be possible. However, it should be noted that while the author takes all possible care to minimize the risk of attack, he is *not* an expert in computer security and that it would certainly be desirable to have such an expert evaluate the code of `gwrdifpk`. Please note also the license conditions of `gwrdifpk`, especially the Disclaimer of Warranty. See [GNU Free Documentation License], page 201.

5.1 Cryptography

Cryptography is an indispensable tool for keeping data secret. It is used by GnuTLS to keep the communication between server and client secret. This is done "behind the scenes", using *X.509 certificates*. See Section 2.4 [X.509 certificates], page 7. That is, `gwrdifpk` simply calls functions from the GnuTLS library and the latter takes care of verifying the certificates and managing the communication.

`gwrdifpk` uses cryptography directly to encrypt and decrypt the "scrambled" iRODS passwords See Section 2.5.1 [Setting up iRODS users], page 7, and Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69. It also uses it for decrypting the passwords for the MySQL databases. Finally, cryptography is also used for encrypting lists of *Transaction Authentication Numbers* (TANs), except that this feature is currently disabled (as of 2013.09.23.). See Section 8.13 [Other user commands], page 95.

`gwrdifpk` uses the GNU Privacy Guard (GPG) for encrypting and decrypting data, listing keys, and for any other purposes requiring cryptography. See Section 2.2 [Prerequisites], page 4. `gpg` (lowercase) is a command-line tool and the GPG package does *not* provide a C API. The libraries that belong to the package are for implementing cryptographic software,

not for making the functionality of `gpg` available to applications. Therefore, in `gwrdifpk`, `gpg` is called in a shell via `popen`[1]

By default, `gwrdifpk` uses an OpenPGP (PGP = Pretty Good Privacy) *secret key without* a passphrase. This is definitely a security risk, but necessary, if it is intended that the server program `gwirdsif` be restarted automatically after having been terminated, intentionally or because of an error. In addition, for testing purposes, it would be very inconvenient to have to type in a passphrase every time the server is started. However, it does mean that `gwrdifpk` is only as secure as the file system on which the secret key is stored: If an attacker gains root privileges, he or she has access to the secret key and can use it for decryption and signing.

However, it is possible to use a secret key with a passphrase. In this case, `gwirdsif` must be invoked with the '`--gpg-passphrase`' argument. The user will then be prompted for the passphrase. `gwrdifpk` includes a shellscript '`start_gwirdsif_with_passphrase.sh`' for invoking `gwirdsif` this way conveniently. See Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69, and Section 33.1 [GPG keys (Shellscripts and Utilities)], page 191.

This is certainly safer than using a secret key without a passphrase, but if `gwirdsif` terminates for any reason, it cannot be started unless the passphrase can be entered somehow, by a person or automatically. However, one would have to ensure that any automatic system would not be vulnerable to an attack similar to the one described above for a secret key without a passphrase, otherwise, there would be no security benefit with respect to the latter.

5.2 Decrypting MySQL passwords

To use an encrypted MySQL password, `gwirdsif` or `gwirdcli` must be invoked using the command-line option '`--mysql-password-filename`'. The username may be specified with the '`--mysql-username`' option, otherwise, the username of the user who invoked the program is used. See Section 6.2.15 [Database options], page 71.

5.3 X.509 Certificates

See Chapter 20 [X.509 Certificate Types], page 134, and Section 28.3 [Certificates database table], page 161.

5.4 Cryptographic operations on iRODS objects

See also Section 8.3 [User commands based on icommands], page 74.

[1] It could also be called via `system`, but as of 2013.09.23., only `popen` is used.

6 Invoking gwirdsif/gwirdcli

`gwirdsif` and `gwirdcli` share the code for processing their command-line options. See Section 26.2 [Process command-line options], page 151.

6.1 Command-line arguments

`remote-hostname`
: For `gwirdcli` only. Required for connections to the server on a remote host, or on the localhost using TLS with or without authentication/authorization with X.509 certificates. If omitted, `gwirdcli` uses a Unix-domain socket to connect with an instance of the server program `gwirdsif` on the localhost.

  For a connection to a server running on the local host, it should be possible on most Unix-like systems to use '`localhost`' as the `remote-hostname` argument, e.g., '`gwirdcli localhost`'.

6.2 Command-line options

Like `setupdbs` (see Section 32.5.1 [Invoking setupdbs], page 186), `gwirdsif` and `gwirdcli` use the `getopt_long_only` function from the GNU C library to parse its command-line arguments. This implies that the options to the programs may be specified using two hyphens, as in the list below, or with a single hyphen. In addition, any option may be abbreviated, as long as the abbreviation is unambiguous. For example, '`--version`' may be specified as '`-version`', '`--ver`' or even '`-v`', since there (currently) are no other options whose names begin with "v". On the other hand, the option '`--sleep-server-enable`' may be abbreviated to '`--sleep-server-e`' but not to to '`--sleep-server`', because the option '`--sleep-server-disable`' also begins with this sequence of characters, making '`--sleep-server`' ambiguous. See Section "Getopt Long Options" in *The GNU C Library Reference Manual*.

6.2.1 Alphabetical list of options

`--anon-port`
: Section 6.2.4 [Connection options], page 67.

`--anonymous`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--bison-trace`
: Section 6.2.8 [Debugging options], page 68.

`--ca-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--cert-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--cert-format`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--client-test-disable`
: Section 6.2.8 [Debugging options], page 68.

`--client-test-enable`
: Section 6.2.8 [Debugging options], page 68.

`--commands`
: Section 6.2.7 [Input/output options], page 67.

`--config-directory`
: Section 6.2.3 [Configuration options], page 67.

`--crl-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--debug-level`
: Section 6.2.8 [Debugging options], page 68.

`--end-server-enable`
: Section 6.2.8 [Debugging options], page 68.

`--error-log-filename`
: Section 6.2.9 [Logging options], page 69.

`--flex-trace`
: Section 6.2.8 [Debugging options], page 68.

`--gpg-key-id`
: Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69.

`--gpg-passphrase`
: Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69.

`--gpg-homedir`
: Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69.

`--help` Section 6.2.2 [Options for getting information], page 67.

`--i-commands`
`--icommands`
: Section 6.2.13 [iRODS options], page 70.

`--input-filename`
: Section 6.2.7 [Input/output options], page 67.

`--irods-server-directory`
: Section 6.2.13 [iRODS options], page 70.

`--jargon-core`
: Section 6.2.13 [iRODS options], page 70.

`--jargon-trunk`
: Section 6.2.13 [iRODS options], page 70.

`--key-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--log-directory`
: Section 6.2.9 [Logging options], page 69.

`--listen-client-port`
: Section 6.2.16 [Pull client options], page 71.

`--log-filename`
: Section 6.2.9 [Logging options], page 69.

`--mysql-password-filename`
: Section 6.2.15 [Database options], page 71.

`--mysql-username`
: Section 6.2.15 [Database options], page 71.

`--no-terminate-on-end-input`
: Section 6.2.8 [Debugging options], page 68.

`--output-filename`
: Section 6.2.7 [Input/output options], page 67.

`--parser-trace`
: Section 6.2.8 [Debugging options], page 68.

`--passphrase-gpg`
: Section 6.2.11 [Options for GPG (GNU Privacy Guard)], page 69.

`--purge-database-interval`
: Section 6.2.10 [Purging options], page 69.

`--purge-database-limit`
: Section 6.2.10 [Purging options], page 69.

`--purge-dc-metadata-interval`
: Section 6.2.10 [Purging options], page 69.

`--purge-dc-metadata-limit`
: Section 6.2.10 [Purging options], page 69.

`--purge-irods-archive-interval`
: Section 6.2.10 [Purging options], page 69.

`--purge-irods-archive-limit`
: Section 6.2.10 [Purging options], page 69.

`--purge-logs-interval`
: Section 6.2.10 [Purging options], page 69.

`--purge-logs-interval-seconds`
: Section 6.2.10 [Purging options], page 69.

`--purge-logs-limit`
: Section 6.2.10 [Purging options], page 69.

`--remote-hostname`
: Section 6.2.4 [Connection options], page 67.

`--save-temp-files`
: Section 6.2.8 [Debugging options], page 68, and Section 6.2.9 [Logging options], page 69.

`--scanner-trace`
: Section 6.2.8 [Debugging options], page 68.

`--server-test-disable`
: Section 6.2.8 [Debugging options], page 68.

`--server-test-enable`
: Section 6.2.8 [Debugging options], page 68.

`--session-id`
: Section 6.2.5 [Session options], page 67.

`--signal-client-disable`
: Section 6.2.6 [Signal options], page 67.

`--signal-client-enable`
: Section 6.2.6 [Signal options], page 67.

`--signal-server-disable`
: Section 6.2.6 [Signal options], page 67.

`--signal-server-enable`
: Section 6.2.6 [Signal options], page 67.

`--sleep-client-disable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-client-enable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-server-disable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-server-enable`
: Section 6.2.8 [Debugging options], page 68.

`--socket-directory`
: Section 6.2.4 [Connection options], page 67.

`--standalone-handle`
: Section 6.2.14 [Handle options], page 71.

`--suppress-prompt`
: Section 6.2.7 [Input/output options], page 67.

`--terminate-on-end-input`
: Section 6.2.7 [Input/output options], page 67.

`--trace` Section 6.2.8 [Debugging options], page 68.

`--version`
: Section 6.2.2 [Options for getting information], page 67.

`--x509-port`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

6.2.2 Options for getting information

`--help` Section 6.2.2 [Options for getting information], page 67.

`--version`
Section 6.2.2 [Options for getting information], page 67.

6.2.3 Configuration options

`--config-directory`

6.2.4 Connection options

`--anon-port`
Section 6.2.4 [Connection options], page 67.

`--remote-hostname`
Section 6.2.4 [Connection options], page 67.

`--socket-directory`
Section 6.2.4 [Connection options], page 67.

6.2.5 Session options

`--session-id`
Section 6.2.5 [Session options], page 67.

6.2.6 Signal options

`--signal-client-disable`
Section 6.2.6 [Signal options], page 67.

`--signal-client-enable`
Section 6.2.6 [Signal options], page 67.

`--signal-server-disable`
Section 6.2.6 [Signal options], page 67.

`--signal-server-enable`
Section 6.2.6 [Signal options], page 67.

6.2.7 Input/output options

`--commands`
Section 6.2.7 [Input/output options], page 67.

`--input-filename`
Section 6.2.7 [Input/output options], page 67.

`--no-terminate-on-end-input`
Section 6.2.8 [Debugging options], page 68, and Section 6.2.7 [Input/output options], page 67.

`--output-filename`
Section 6.2.7 [Input/output options], page 67.

`--suppress-prompt`
: Section 6.2.7 [Input/output options], page 67.

`--terminate-on-end-input`
: Section 6.2.7 [Input/output options], page 67.

6.2.8 Debugging options

`--bison-trace`
: Section 6.2.8 [Debugging options], page 68.

`--client-test-disable`
: Section 6.2.8 [Debugging options], page 68.

`--client-test-enable`
: Section 6.2.8 [Debugging options], page 68.

`--debug-level`
: Section 6.2.8 [Debugging options], page 68.

`--end-server-enable`
: Section 6.2.8 [Debugging options], page 68.

`--flex-trace`
: Section 6.2.8 [Debugging options], page 68.

`--no-terminate-on-end-input`
: Section 6.2.8 [Debugging options], page 68.

`--parser-trace`
: Section 6.2.8 [Debugging options], page 68.

`--save-temp-files`
: Section 6.2.8 [Debugging options], page 68, and Section 6.2.9 [Logging options], page 69.

`--scanner-trace`
: Section 6.2.8 [Debugging options], page 68.

`--server-test-disable`
: Section 6.2.8 [Debugging options], page 68.

`--server-test-enable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-client-disable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-client-enable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-server-disable`
: Section 6.2.8 [Debugging options], page 68.

`--sleep-server-enable`
: Section 6.2.8 [Debugging options], page 68.

`--trace` Section 6.2.8 [Debugging options], page 68.

6.2.9 Logging options

`--error-log-filename`
: Section 6.2.9 [Logging options], page 69.

`--log-directory`
: Section 6.2.9 [Logging options], page 69.

`--log-filename`
: Section 6.2.9 [Logging options], page 69.

`--save-temp-files`
: Section 6.2.8 [Debugging options], page 68, and Section 6.2.9 [Logging options], page 69.

6.2.10 Purging options

`--purge-database-interval`
: Section 11.2 [Purging (gwirdsif)], page 99, and Section 12.1 [Purging (gwirdcli)], page 100

`--purge-database-limit`
: Section 11.2 [Purging (gwirdsif)], page 99, and Section 12.1 [Purging (gwirdcli)], page 100

`--purge-irods-archive-interval`
: Section 11.2 [Purging (gwirdsif)], page 99.

`--purge-irods-archive-limit`
: Section 11.2 [Purging (gwirdsif)], page 99.

`--purge-logs-interval`
: Section 11.2 [Purging (gwirdsif)], page 99, and Section 12.1 [Purging (gwirdcli)], page 100

`--purge-logs-interval-seconds`
: Section 11.2 [Purging (gwirdsif)], page 99, and Section 12.1 [Purging (gwirdcli)], page 100

`--purge-logs-limit`
: Section 11.2 [Purging (gwirdsif)], page 99, and Section 12.1 [Purging (gwirdcli)], page 100

`--purge-dc-metadata-interval`
: Section 11.2 [Purging (gwirdsif)], page 99.

`--purge-dc-metadata-limit`
: Section 11.2 [Purging (gwirdsif)], page 99.

6.2.11 Options for GPG (GNU Privacy Guard)

`--gpg-key-id ARG`
: Required by `gwirdsif`. Not used by other programs, but available in the other programs that link with `cmdlnopt.o` (such as `gwirdcli` and `gwrdwbap`). '`ARG`' is the GPG key ID of the key pair used used for encrypting and decrypting the

"scrambled" iRODS passwords, and possibly for other purposes. See Chapter 5 [Security considerations], page 61.

`--gpg-passphrase`
`--passphrase-gpg`
: These options are synonyms. No argument. Currently only used by `gwirdsif`, but available in the other programs, as described for '`--gpg-key-id`', above.

  '`--gpg-passphrase`' is only used when the GPG key pair used for encrypting and decrypting the "scrambled" iRODS passwords has a passphrase. See Section 2.5.1 [Setting up iRODS users], page 7.

  If this option is used, the user will be prompted for the passphrase. This is certainly safer than using a GPG key without one, but much less convenient. For one thing, it means that `gwirdsif` cannot be started unattended, for example, by a `cron` job, if it exited for some reason.

  For another thing, this feature causes problems when `gwirdsif` is called directly from within a `Makefile` (or indirectly from a `Makefile.am` file). Therefore, `gwrdifpk` includes shellscripts which can be called from them instead, when using '`--gpg-passphrase`'. See Section 33.1 [GPG keys (Shellscripts and Utilities)], page 191.

  See also Chapter 5 [Security considerations], page 61, for more information.

`--gpg-homedir ARG`
: Set the directory containing the GPG configuration files. The default is '`$HOME/.gnupg`', i.e., the '`.gnupg/`' directory below the user's home directory.

6.2.12 Options for X.509 Authentication/Authorization

`--anonymous`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--ca-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--cert-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--cert-format`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--crl-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--key-filename`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

`--x509-port`
: Section 6.2.12 [Options for X.509 Authentication/Authorization], page 70.

6.2.13 iRODS options

`--i-commands`
`--icommands`
: Section 6.2.13 [iRODS options], page 70.

`--irods-server-directory`
: Section 6.2.13 [iRODS options], page 70.

`--jargon-core`
: Section 6.2.13 [iRODS options], page 70.

`--jargon-trunk`
: Section 6.2.13 [iRODS options], page 70.

6.2.14 Handle options

`--standalone-handle`
: Section 6.2.14 [Handle options], page 71.

6.2.15 Database options

`--mysql-username USERNAME`
: '`USERNAME`' is the MySQL username used for accessing the databases.

`--mysql-password-filename [FILENAME]`
: If specified, '`FILENAME`' is the name of the file containing the encrypted MySQL password for the user specified with the '`--mysql-username`' option (see above). Otherwise, the value of `DEFAULT_MYSQL_PASSWORD_FILENAME`, namely '`mysql_password.gpg.asc`' is used instead. See Section 13.1 [Global constants], page 101.

6.2.16 Pull client options

`--listen-client-port`

7 Pull archiving

Pull archiving refers to the procedure where the server program `gwirdsif` contacts the *pull client* `gwirdpcl` with one or more *pull requests*. `gwirdpcl` then calls a version of the "normal" client function to send changed files to `gwirdsif`. The pull client thus acts as a server with the "server" program `gwirdsif` acting as a client with respect to `gwirdpcl`.

See also Section 8.12 [Pull requests], page 95, Section 9.1 [Pull responses], page 96, Section 6.2.16 [Pull client options], page 71, Chapter 23 [class Pull_Request_Type], page 145, Chapter 24 [class Pull_Response_Type], page 147, Section 28.15 [Pull Request database table (gwirdsif)], page 172, and Section 29.7 [Pull Response database tables (gwirdcli)], page 175.

8 User commands

8.1 Conventions

`keyword` *Keywords* are formatted in a typewriter face. They may be the command names, such as `put` and `get`, or modifiers (a.k.a. *option keywords*, see Section 8.2 [Common options], page 73, below), such as `remote_filename` or `generate`. Keywords may be typed in all lowercase or all uppercase letters, e.g., `add_metadata` or `ADD_METADATA`, but not `Add_Metadata` or `Add_MetaData`.

FLAGS A single hyphen followed by a single letter or two hyphens followed by at least one letter, e.g., '`-f`', '`--force`', '`-h`' or '`--help`'. For user commands based on icommands, *FLAGS* are usually options for the icommand and are passed to it without processing by `gwirdsif`. In some cases, options for icommands may not make sense when using `gwrdifpk`. See Section 8.3 [User commands based on icommands], page 74.

For other commands, *FLAGS* may have nothing to do with icommands or iRODS.

`<...>` String. Strings may be *delimited* by quotation marks (`"..."`) or *undelimited*.

An undelimited string may contain alphanumerical characters and various punctuation and special characters. The first character must be an alphanumerical character or one of the following characters: `./;~@()+\?$`. It may be followed by zero or more characters, which must either be alphanumerical or one of the following characters: `./;-_~+@()\?~$`. That is, '`-`' and '`_`' may occur in the string, but not as the first character. **Please note** that '`:`' may not occur as the first character in an undelimited string! A *time specification* may begin with a leading colon. See Section 8.2.1 [Delay], page 74, below.[1]

These rules imply that a delimited string may be empty, i.e., `""`, whereas an undelimited string may not.

These are the rules as of 2013.10.23. They are subject to change. See the corresponding *scanner rule* in '`scanner.web`' for the current definition.

[...] Optional item.

`SAMPLE_COMMAND_NAME` [*FLAGS*] <*filename*> [<*remote filename*>] [Command]

8.2 Common options

Option keywords may be typed in all lowercase or all uppercase, e.g., '`delay`' or '`DELAY`', but not '`Delay`' or '`DeLay`'. Spaces preceding or following an equals sign ('`=`') are optional, e.g., '`delay=21`' and '`delay = 21`' are equivalent.

[1] It would be possible to distinguish between an undelimited string with a leading colon and a time specification, as long as the former doesn't only contain digits and colons, but the author sees no need to implement this at the present time.

8.2.1 Delay

`no_delay` The command is executed immediately, before any other commands from the input are read.

`delay`
`delay INTEGER`
`delay = INTEGER`
`delay <time specification>`
`delay = <time specification>`
: The command is processed in the normal way, after any commands preceding it in the input. If a delay is specified, either with an `INTEGER` (for the number of seconds) or a <time specification>, then execution *may* be delayed by this additional amount. **Please note**: For some commands, the additional delay may not have been implemented. In some cases, it may not make sense and will not be implemented in the future, either.

A <time specification> consists of integers separated by colons, whereby there must be at least one and no more than four integers, colons may appear next to each other (i.e., an integer may be missing), and there may be a trailing colon. For example, the following are valid time specifications:

`1:2:3:4` 1 day, 2 hours, 3 minutes, 4 seconds.

`1:2:3:4:` As above.

`::2` Two hours.

`::2::3`
`::2::3:` Two hours, 3 seconds.

`::::10` Ten seconds.

8.3 User commands based on icommands

`pwd` [Command]
The server sends the current server-side iRODS directory to the client. For example:

(Client output:)
```
pwd -->
/tempZone/home/lfinsto
```

`cd` [<*directory name*>] [Command]
Change the *current working directory*. If <*directory name*> is omitted, change to the user's *home directory*.

`mkdir` [*FLAGS* . . .] *DIRECTORY-NAME* [. . .] [Command]
Create a directory (i.e., an *iRODS collection*).

`ls` [*FLAG* . . .] [*PATH* . . .] [*delay option*] [Command]
FLAG may be any flag valid for the '`ils`' icommand. *PATH* may refer to an iRODS data object or a collection. **Please note**: An error occurs if multiple data objects are specified together with the '`-l`' or '`-L`' flags. This appears to be a bug in the

implementation of the icommand. This error doesn't occur when multiple collections are specified, or multiple collections with a single data object (as long as all of the objects exist).

<delay option>: If a delay is specified, commands following this command in the input are executed first. This may be useful if `put` commands, or other commands that result in a dialogue between the client and the server, *precede* this command in the input. Specifying a delay ensures that any such dialogues complete before the `ls` command is executed. See Section 8.2.1 [Delay], page 74.

See also Section 14.2 [Scan_Parse_Parameter_Type Member Functions], page 111.

`mv` [*FLAG ...*] <*old filename or path*> <*new filename or path*> [Command]

FLAG may be any flag valid for the '`imv`' icommand. <new filename or path> may refer to an iRODS data object or a collection.

See also Section 14.2 [Scan_Parse_Parameter_Type Member Functions], page 111.

`put` [*FLAGS*] [(*PID option* | *cryptographic option* | *compression option*) ...]
<*local filename*> [`remote_filename` <*path*>] [Command]

Transfer *local filename* to the server and store it in the server-side iRODS system.

If *local filename* is a directory, a *compression option* must be used. In this case, `tar` is used to store the contents of the directory in an archive, which is then compressed using either `gzip` (the default) or `bzip2`. If *local filename* is a directory and no compression option is specified, an error message is issued and *local filename* is not sent to the server.

FLAGS:

`-f`, `--force`
: Transfer and store the file, overwriting any existing server-side iRODS object of the same name and path.

`gwirdsif` simply passes any other flags to the icommand `iput`.

PID options:

`pid` [<*string*>]
: Generate a PID (*persistent identifier*), a.k.a. *handle*. If <*string*> is provided, it should be used as the handle. It must be a valid handle and not already exist. Otherwise, the server sends an error message to the client.

`generate`
: Generate a handle. In this case, the server program chooses a name for the handle. Handles created in this way are numbered consecutively.

`prefix` <*string*>
: Use the prefix <*string*> for the handle. It must be a prefix for which the server-side handle service is responsible. If not specified, the *default prefix* for the user is used. See Section 14.1.3 [Variables (Scan_Parse_Parameter_Type)], page 109.

`suffix` <*string*>
: <*string*> is appended to the handle.

`institute` <*string*>
: <*string*> is included within the handle, before the suffix, if any.

See Section 1.2 [Handles], page 2.

Cryptographic options:

`encrypt` Encrypt the file before sending it to the server.

`sign` Sign the file. This option only takes effect if the `encrypt` is also specified.

`clearsign`
: Clearsign the file. This option implies that the file is *not* encrypted.

`detached` Create a detached signature. It will be sent to the server and stored in an iRODS object.

`verify` Verify the signature on the server-side. In this case, the user's *public key* must have been stored on the server-side.

See Section 5.4 [Cryptographic operations on iRODS objects], page 62.

Compression options:

`compress`
`compress gzip`
: Use `gzip` to compress *local filename* or, if *local filename* is a directory, the archive file created from it using `tar`. The compressed file is then sent to the server.

`compress bzip2`
: As above, except that `bzip2` is used for compression instead of `gzip`.

Example:

```
put -f +pid +gen "abc.txt" remote_filename "subdir_1/"
```

Client output:

```
put -->
Filename:     /tempZone/home/lfinsto/subdir_1/abc.txt
Exit status:  0
Response:     'iput' command succeeded, returning 0

put -->
Filename:     /tempZone/home/lfinsto/subdir_1/abc.txt
Exit status:  0
Response:     Success:  Generated PID '12345/00002'

put -->
Filename:     /tempZone/home/lfinsto/subdir_1/abc.txt
Exit status:  0
Response:     Added handle values with type == 'IRODS_OBJECT' \
                 and type == 'CREATOR_INDEX' successfully
```

```
put -->
Filename:     /tempZone/home/lfinsto/subdir_1/abc.txt
Exit status:  0
Response:     Success:  Stored PID '12345/00002' in iRODS object \
                 metadata
```

`get` <*filename*> [*FLAGS*] [*OPTION ...*][`local_filename` <*filename*>] [Command]

The only *FLAG* processed specially by `gwirdsif` is '`-f`' for "force". If used, the client-side file <*filename*> will be overwritten, if it exists.[2] Other flags are passed to '`iget`' unexamined and may not make sense when accessing the iRODS server remotely via `gwrdifpk`.

Options:

`decrypt` If *filename* is encrypted, decrypt it. Decryption occurs on the client-side.

`verify` If *filename* is signed, verify the signature. Verification occurs on the client-side. Please note that if the signature is *not* detached, *filename* GPG must decrypt it in order to verify the signature. The private key needed for decryption and its passphrase must therefore be available on the client-side. However, the decrypted text will only be stored in a file if the `decrypt` option is also specified. If, however, *filename* has been *clearsigned* or it has *not* been encrypted and the signature is detached, it may be verified using the public key from the key pair used for signing.

`store-signature`
If *filename* is signed with a detached signature, store the latter in a file.

`expand` If *filename* has been compressed, expand it. If it is a compressed tar file, the contents will be extracted.

See Section 5.4 [Cryptographic operations on iRODS objects], page 62.

Examples:

```
get "abc.txt"
```

Client output:

```
get -->
Local filename:  abc.txt
Response code:   2
Response:        Success.  Queuing "SEND FILE" command.

get -->
Remote filename: /tempZone/home/lfinsto/abc.txt
Local filename:  abc.txt
```

[2] Please note that the '`-f`' option is always used when the icommand '`iget`' is called on the server-side. This is because the iRODS object is written to a temporary file, which must be created before the call to '`iget`'.

```
Exit status:     0
Overwrite:       False

Received remote file ‘/tempZone/home/lfinsto/abc.txt’.
Stored in local file ‘abc.txt’.
```

If the file ‘abc.txt’ already exists in the current working directory on the client-side, the server will retrieve the iRODS object from the iRODS server and send it to the client, but storing it on the client-side will fail. However, the client will create a temporary file for the contents of the iRODS object and issue a message with the path of the temporary file:

```
get "abc.txt"
```

Client output:

```
get -->
Local filename:  abc.txt
Response code:   2
Response:        Success.  Queuing "SEND FILE" command.

get -->
Remote filename: /tempZone/home/lfinsto/abc.txt
Local filename:  abc.txt
Exit status:     0
Overwrite:       False

[Thread 0] WARNING!  In ‘Scan_Parse_Parameter_Type::receive_file’:
File ‘/home/lfinsto/abc.txt’ already exists and ‘overwrite’ == ‘false’.
Setting ‘discard’ == ‘true’.
[Thread 0] In ‘Scan_Parse_Parameter_Type::receive_file’:
‘discard’ == 1
‘gen_temp_file’ == 0
Wrote file contents sent from server to temporary file \
‘/tmp/gwirdcli.u0EypV’.
Exiting function unsuccessfully with return value 2.
[Thread 0] ERROR!  In ‘zzparse’, rule
‘statement: GET_ZZ FILE_ZZ RESPONSE_ZZ STRING_ZZ INTEGER_ZZ STRING_ZZ’:
‘Scan_Parse_Parameter_Type::receive_file’ failed, returning 2.
Failed to receive file ‘/tempZone/home/lfinsto/abc.txt’ \
or store it in ‘abc.txt’.
Local file already exists and ‘overwrite’ == false’
Stored file contents in file:  /tmp/gwirdcli.u0EypV
Will try to continue.
```

Using the ‘-f’ (“force”) flag solves this problem:

```
get -f "abc.txt"
```

Client output:

```
get -->
Local filename:  abc.txt
Response code:   2
Response:        Success.  Queuing "SEND FILE" command.

get -->
Remote filename: /tempZone/home/lfinsto/abc.txt
Local filename:  abc.txt
Exit status:     0
Overwrite:       True

Received remote file ‘/tempZone/home/lfinsto/abc.txt’.
Stored in local file ‘abc.txt’.
```

`rm` [*FLAGS*] [*options*] <*PATH*> [...] [Command]

Delete the iRODS objects specified by the *PATH*(s).

Options:

`database` Delete data pertaining to the *PATH*(s) from the various databases in addition to deleting the iRODS object itself.

`database_only`
Delete only data pertaining to the *PATH*(s) from the various databases without deleting the iRODS object itself.

delay option
By default, the iRODs object is deleted immediately. If a delay is specified, the iRODS object will instead be *marked for deletion* and be deleted *at the earliest* at the end of the period specified. See Section 8.2.1 [Delay], page 74, and Section 11.2 [Purging], page 99.

8.4 iRODS objects

`undelete` [*OPTIONS* ...] <*filename*> ... [Command]

Unmark the iRODS object or objects named by the <*filename*> argument or arguments for deletion. For this to work, they must have been *marked for deletion* by using the `rm` command with the ‘`delay`’ option. See Section 8.3 [User commands based on icommands], page 74.

If a <*filename*> does not exist or has not been marked for deletion, the client prints an error message on the standard error output.

Options:

`database` Entries in the `gwirdsif` database containing information pertaining to the iRODS object or objects will be *unmarked for deletion*.

`database_only`
Only entries in the `gwirdsif` database containing information pertaining to the iRODS object or objects will be *unmarked for deletion*, the iRODS object or objects will *not* be unmarked for deletion.

8.5 Handles

`create handle` [*ARGUMENT* ...] [Command]

Create a *handle*, a.k.a. *persistent identifier* or *PID*.

With no arguments, a handle is created with a name chosen by `gwirdsif`. Such names consist of the user's default prefix followed by at least five hexadecimal digits and are created in numerical order, e.g., '`12345/00001`' would be the first handle created in such a way, followed by '`12345/00002`', '`12345/00003`' ... '`12345/0000A`', '`12345/0000B`', and so on.

Arguments may be used to specify another name for the handle as well as the contents of the `idx`, `type` and `data` fields:

<*string*>
`pid` <*string*>
: <*string*> specifies the "name" of the handle. These arguments are equivalent, that is, the keyword '`pid`' is "syntactic sugar".

`idx` <*integer*>
: Specify the index of the handle.

`type` <*string*>
: Specify the type of the handle.

`data` <*string*>
: Specify the contents of the `data` field of the handle.

`delete handle` [*OPTION* ...] <*string*> [Command]

Mark the handle named *string* for deletion.

Options:

`immediate`
: Mark the handle for immediate deletion. See Section 11.2 [Purging gwirdsif], page 99.

<*delay option*>
: The handle will be marked for deletion with the delay specified. See Section 8.2.1 [Delay], page 74.

`undelete handle` <*string*> [Command]

`get handle pid` <*string*> [*OPTION* ...] [Command]

Retrieve the handle values for the PID <*string*>.

Options:

`no-store` Don't store the handle values in the client-side `gwirdcli.handles` database table.

Example:

```
get handle pid "12345/00001"
```

Client output:

```
get handle -->
Response code:   0
filename:        /tempZone/home/lfinsto/abc.txt
handle:          12345/00001
idx:             1
type:            IRODS_OBJECT
data_length:     30
data:            /tempZone/home/lfinsto/abc.txt
ttl_type:        0
ttl:             86400
timestamp:       1364483174 (Thu, 2013-03-28 15:06:14 UTC)
refs_length:     0
admin_read:      1
admin_write:     1
pub_read:        1
pub_write:       0
handle_id:       50
handle_value_id: 112
created:         1364479574 (Thu, 2013-03-28 14:06:14 UTC)
last_modified:   0
created_by_user: 1

get handle -->
Response code:   0
filename:
handle:          12345/00001
idx:             300
type:            HS_ADMIN
data_length:     22
data:            (binary)
ttl_type:        0
ttl:             86400
timestamp:       1364483174 (Thu, 2013-03-28 15:06:14 UTC)
refs_length:     0
admin_read:      1
admin_write:     1
pub_read:        1
pub_write:       0
handle_id:       50
handle_value_id: 111
created:         1364479574 (Thu, 2013-03-28 14:06:14 UTC)
last_modified:   0
created_by_user: 1
```

`get handle file` <*filename*> [*OPTION ...*] [Command]

Retrieve the handle values for the file <*filename*>.

Options:

`no-store` Don't store the handle values in the client-side `gwirdcli.handles` database table.

Example:

```
get handle file "abc.txt"
```

Client output:

```
get handle -->
Response code:   0
filename:        abc.txt
handle:          12345/00001
idx:             1
type:            IRODS_OBJECT
data_length:     30
data:            /tempZone/home/lfinsto/abc.txt
ttl_type:        0
ttl:             86400
timestamp:       1364483174 (Thu, 2013-03-28 15:06:14 UTC)
refs_length:     0
admin_read:      1
admin_write:     1
pub_read:        1
pub_write:       0
handle_id:       50
handle_value_id: 112
created:         1364479574 (Thu, 2013-03-28 14:06:14 UTC)
last_modified:   0
created_by_user: 1

get handle -->
Response code:   0
filename:        abc.txt
handle:          12345/00001
idx:             300
type:            HS_ADMIN
data_length:     22
data:            (binary)
ttl_type:        0
ttl:             86400
timestamp:       1364483174 (Thu, 2013-03-28 15:06:14 UTC)
refs_length:     0
admin_read:      1
admin_write:     1
pub_read:        1
pub_write:       0
```

```
handle_id:       50
handle_value_id: 111
created:         1364479574 (Thu, 2013-03-28 14:06:14 UTC)
last_modified:   0
created_by_user: 1
```

The output for this command is similar to that for the `get handle pid` <*string*> command above, except for the value of the `filename` field. In the case of this command, the user passes the filename to the command, so it is known and only handle values are retrieved that correspond to this file. With `get handle pid` <*string*>, the filename is only known when the handle value is of an appropriate type (e.g., '`IRODS_OBJECT`', as in the previous example), in which case the filename is stored in the '`data`' field of the handle value. Please note that a given handle may have multiple handle values referring to the same or different files.

8.5.1 Handle values

`add handle_value` [*ARGUMENT ...*] [Command]

Add a *handle value* to an existing handle. The arguments are as for the `create handle` command (see above).

`delete handle_value` [*OPTION ...*] <*string*> [Commands]
`delete handle_values` [*OPTION ...*] <*string*>
`delete handle_value` [*OPTION ...*] <*handle value specification*>
`delete handle_values` [*OPTION ...*] <*handle value specification*>

Mark the handle value or values specified with <*string*> or <*handle value specification*> for deletion.

A *handle value specification* is in effect an *undelimited string* specifying a handle value, e.g., '`12345/00001:3`' or '`12345/00001:TYPE`'. See Section 8.1 [Conventions], page 73, above. It consists of five parts:

1. the *handle prefix*, which must be exactly five digits
2. a slash ('`/`'),
3. the handle itself, i.e., at least one character which must be alphanumerical, a hyphen ('`-`') or underscore ('`_`').
4. a colon
5. the index or type of the the handle, i.e., the contents of the handle's `idx` or `type` fields. A number is interpreted as the index. Otherwise, the same characters may appear in this part as for the handle itself (see above).

The options are as for the `delete handle` command (see above).

Options:

`immediate`
: Mark the handle value or values for immediate deletion. See Section 11.2 [Purging gwirdsif], page 99.

<*delay option*>
: The handle value or values will be marked for deletion with the delay specified. See Section 8.2.1 [Delay], page 74.

`undelete handle_value` <*string*> [Commands]
`undelete handle_values` <*string*>
`undelete handle_value` <*handle value specification*>
`undelete handle_values` <*handle value specification*>
Unmark the handle value or values specified with <*string*> or <*handle value specification*> for deletion.

8.6 Dublin Core metadata

`add metadata` <*metadata filename*> <*iRODS object path*> [Command]
[*OPTION* . . .]
Transfer the contents of the file *metadata filename* to the server and associate it with the iRODS object <iRODS object path>.

Options:

`force_add`
Add the Dublin Core metadata even if the iRODS object doesn't exist.

`store`
`force_store`
Store the Dublin Core metadata in an iRODS object of its own. If '`force_store`' is specified, do this even if the iRODS object doesn't exist.

`force`
`force_all`
Equivalent to '`force_add`' and '`force_store`'.

`get metadata` <*path*> [*OPTION* . . .] [Command]
Retrieve the Dublin Core metadata for the iRODS object *path*. Other data pertaining to the iRODS object is also sent to the client. The Dublin Core metadata is stored in a temporary file.

`show` Show the Dublin Core metadata. This is done by generating a `show metadata` command (see below).

`store` Store the Dublin Core metadata on the client side.

`output` Write the file of Dublin Core metadata to written to standard output.

`get metadata` *INTEGER* [. . .] [*OPTION* . . .] [Command]
Retrieve the Dublin Core metadata where the `INTEGER` values refer to the `dublin_core_metadata_id` field of rows in the server-side `gwirdsif.Dublin_Core_Metadata` database table.

This commmand takes the same options as the version with the `<path>` argument, above.

`get metadata pid` <*string*> [output] [Command]
Retrieve the Dublin Core metadata for the iRODS object that has the handle *string*.
Please note: As of 2013.10.24., this command exists, but is not yet functional.

`delete metadata` <*iRODS object path*> [...] [*OPTION* ...] [Command]

Mark the Dublin Core metadata for the iRODS object or objects named by <iRODS object path> for deletion. With no options, the corresponding entry or entries in the `gwirdsif.Dublin_Core_Metadata` database table are marked for deletion.

For the sake of simplicity, the following descriptions of the options assume a single <iRODS object path>. However, they function analogously for calls with more than one <iRODS object path>.

Options:

`file` If the metadata had been stored in an iRODS object of its own, i.e., if the `add metadata` had been called with the '`store`' option, mark the iRODS object for deletion.

`file_only`
Mark only the iRODS object in which the Dublin Core metadata has been stored, for deletion, if it exists. Do not mark the entry in the `gwirdsif.Dublin_Core_Metadata` database table for deletion.

`immediate`
Mark the Dublin Core metadata and/or the iRODS object for immediate deletion. In this case, the deletion timestamp or timestamps are set to a value 366 days in the past and the thread running the function `purge_dc_metadata` is "woken up". See Section 26.10 [Deleting and rotating files], page 154.

`delay` [[=] (`INTEGER` | <*time specification*>)]

`no_delay` See Section 8.2.1 [Delay], page 74.

`force` Mark for deletion even if <iRODS object path> has already been marked for deletion. Without this option, if <iRODS object path> is already marked for deletion, it is not remarked and a warning is sent to the client. This option can be used to remark an <iRODS object path> for deletion using different options.

`save_db_entry`
If this option is used together with the '`file`' or '`file_only`' option, then only the actual iRODS object will be deleted, while the corresponding entry in the `gwirdsif.Irods_Objects` database table will be saved.

If this option is used without either the '`file`' or '`file_only`' option, it has no effect and is ignored.

Example:

```
delete metadata abc.txt
```

Client output:

```
Delete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Options:                      0 (00000000)
Message:                      Success
```

`undelete metadata` <*iRODS object path*> [...] [Command]

Unmark the Dublin Core metadata for the iRODS object or objects named by the occurrences of <iRODS object path> for deletion.

```
undelete metadata abc.txt
```

Client output:

```
Undelete metadata value response -->
Response code:                0 (GW_SUCCESS)
Filename (iRODS object path): /tempZone/home/lfinsto/abc.txt
Message:                      Success
```

`show metadata` [*OPTION* ...] [Command]

Retrieve Dublin Core metadata from the server-side `gwirdsif.Dublin_Core_Metadata` database table. See Section 28.13 [Dublin Core database tables (gwirdsif)], page 170.

Unlike the `get metadata` command described above, this command does not take an iRODS object path as an argument and does not send information about iROD objects to the client. Nor does it send the Dublin Core metadata in the form of a file. Instead, it sends the values of the fields from the `gwirdsif.Dublin_Core_Metadata` and optionally the `gwirdsif.Dublin_Core_Metadata_Sub` database tables individually. In the client-side parser, one or more objects of type `class Dublin_Core_Metadata_Type` are created and "shown", i.e., the member function `Dublin_Core_Metadata_Type::show` is called on them. See Section 21.1.2 [Member Functions (Dublin_Core_Metadata_Type)], page 140.

TODO: It would also be possible to use them to store the Dublin Core in a client-side database table. However, it would make more sense to use the `get metadata` command for this purpose, which would require modifying the latter command.

Options:

`INTEGER` Refers to the `dublin_core_metadata_id` field of the entries in the `gwirdsif.Dublin_Core_Metadata` database table.

`full` Send the corresponding entries from the `gwirdsif.Dublin_Core_Metadata_Sub` database table.

`store` Store the Dublin Core metadata in the client-side `gwirdcli` database.

`user`
`group`
`all` These options are not yet functional (as of 2013.12.17.).

8.7 Retrieving information

`whoami` [Command]

Sends brief user information to the client.

Example:

```
whoami
```

Client output:

```
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston
```

`get_user_info` [<*username*> [*delay option*]] [Commands]

Sends detailed user information to the client.

Example:

```
get_user_info
```

Client output:

```
Get user info response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
distinguished_name:
     organization:.................GWDG
     organizationalUnitName:......gwrdifpk
     commonName:...................Laurence Finston
     countryName:.................DE
     localityName:................Goettingen
     stateOrProvinceName:.........Germany
     user_id:.....................1
     user_name:...................lfinsto
privileges:
   superuser:                1
   delegate:                 1
   delete_handles:           1
   show_user_info:           1
   show_certificates:        1
   show_distinguished_names: 1
   show_privileges:          1
irods_password_encrypted:
irods_password_encrypted_timestamp:
irods_homedir:                        /tempZone/home/lfinsto
irods_current_dir:                    /tempZone/home/lfinsto
irods_zone:                           tempZone
irods_default_resource:               demoResc
irods_env_filename:
```

```
irods_auth_filename:
default_handle_prefix_id:            1
default_handle_prefix:               12345
default_institute_id:                1
default_institute_name:              GWDG Test Institute
public_key_id:
certificate:
     serialNumber:.............2 (hexadecimal)
     organization:.............GWDG
     organizationalUnitName:......gwrdifpk
     commonName:...............Laurence Finston
     countryName:..............DE
     localityName:.............Goettingen
     stateOrProvinceName:.....Germany
     user_id:..................1
     certificate_id:...........2
     issuer_cert_id:...........0
     user_name:................lfinsto
     is_ca:....................0
     is_proxy:.................0
     Activation time:..........2013-05-03 13:02:53 UTC
     Expiration time:..........2033-04-28 13:02:56 UTC
```

Users with the `show user info` privilege may call this command with the <username> argument:

```
whoami
get_user_info jsmith
```

Client output:

```
Whoami response -->
Response code:  0
User Info:
user_id:      1
username:     lfinsto
Common Name:  Laurence Finston

Get user info response -->
Response code:  0
User Info:
user_id:      3
username:     jsmith
distinguished_name:
     organization:.............GWDG
     organizationalUnitName:......gwrdifpk
     commonName:...............Jane Smith
     countryName:..............DE
```

```
     localityName:.............Goettingen
     stateOrProvinceName:.....Niedersachsen
     user_id:..................3
     user_name:................jsmith
privileges:
   superuser:               0
   delegate:                0
   delete_handles:          0
   show_user_info:          0
   show_certificates:       0
   show_distinguished_names: 0
   show_privileges:         0
irods_password_encrypted:
irods_password_encrypted_timestamp:
irods_homedir:                      /tempZone/home/jsmith
irods_current_dir:
irods_zone:                         tempZone
irods_default_resource:             demoResc
irods_env_filename:
irods_auth_filename:
default_handle_prefix_id:           1
default_handle_prefix:              12345
default_institute_id:               1
default_institute_name:             GWDG Test Institute
public_key_id:
certificate:
     serialNumber:.............6 (hexadecimal)
     organization:.............GWDG
     organizationalUnitName:......gwrdifpk
     commonName:...............Jane Smith
     countryName:..............DE
     localityName:.............Goettingen
     stateOrProvinceName:.....Niedersachsen
     user_id:..................3
     certificate_id:...........4
     issuer_cert_id:...........1
     user_name:................jsmith
     is_ca:....................0
     is_proxy:.................0
     Activation time:..........2013-05-15 09:08:07 UTC
     Expiration time:..........2033-05-10 09:08:10 UTC
```

If a user without the `show user info` privilege tries this, an error message is issued:

```
whoami
get_user_info jsmith
```

Client output:

```
Whoami response -->
Response code:  0
User Info:
user_id:      2
username:     jdoe
Common Name:  John Doe

Get user info response -->
Response code:  1
Server-side error:  "get_user_info" command failed.
```

`show certificate [user]` [Commands]
`show certificates [all]`

The server sends information about the X.509 certificate or certificates to the client, which prints it to standard output.

Which certificates may be shown depends on whether the user has the `show_certificates` privilege. See Section 28.8 [Privileges database table], page 164. Any user can use this command to retrieve information about his or her own certificate. Users with the `show_privileges` may retrieve information about other users' certificates as well.

When *group management* is implemented, it should be possible to exercise finer control over the way certificates are shown on the basis of *roles* within groups. See Section 28.4 [Groups database tables and views], page 162.

Options:

`user`
`all`
: Currently, these options have no effect. The `show certificate` command shows the current user's certificate while the `show certificates` command shows the certificates for all users, if the current user has the `show_certificates` privilege (see above).

  In the future, additional options and/or arguments should make it possible to specify the certificate or certificates to be shown more precisely, e.g., `show certificate [user [<`*string*`>]]`.

`show groups all` [Command]

The server sends information about all existing groups to the client, which prints it to standard output. The current user must have the `show_groups` privilege. Otherwise, an error message is output. See Section 28.8 [Privileges database table], page 164.

Example:

```
show groups all
```

Client output:

```
Show groups response -->
Response code:  0
Group info for 2 groups:
```

```
Group_Type:
group_id ==           1
‘group_name’ ==       test_group_0
‘creator_id’ ==       1
‘creator_username’ == lfinsto
‘created’ ==          1370433954 == 2013-06-05 14:05:54
member_id_map.size() == 2
member_id_map:
User ID:  1  Username: lfinsto   Privileges: 7 (octal)
   Add user privilege:      true
   Delete user privilege:   true
   Delete group privilege:  true
User ID:  2  Username: jdoe      Privileges: 0 (octal)
   Add user privilege:      false
   Delete user privilege:   false
   Delete group privilege:  false

Group_Type:
group_id ==           2
‘group_name’ ==       test_group_1
‘creator_id’ ==       1
‘creator_username’ == lfinsto
‘created’ ==          1370433954 == 2013-06-05 14:05:54
member_id_map.size() == 2
member_id_map:
User ID:  1  Username: lfinsto   Privileges: 6 (octal)
   Add user privilege:      false
   Delete user privilege:   true
   Delete group privilege:  true
User ID:  2  Username: jdoe      Privileges: 1 (octal)
   Add user privilege:      true
   Delete user privilege:   false
   Delete group privilege:  false
```

8.8 Testing gwirdsif

`distinguished_name` <*distinguished name*> [Command]

`[STRING]` is the *distinguished name* from an X.509 certificate, for example: `"/C=DE/O=GridGermany/OU=Gesellschaft fuer wissenschaftliche Datenverarbeitung mbH/CN=Laurence Finston"`.

This command is used only when the client contacts a server running locally through a Unix-domain socket or via GnuTLS with the ‘`--anonymous`’ option. See Section 4.2.1 [Invoking gwirdcli (Getting Started)], page 16.

`sleep` *INTEGER* [Commands]
`sleep client` *INTEGER*

These commands cause the server or the client to "sleep" for *INTEGER* seconds. That is, the server or client calls the C library function `sleep`. In the case of the server, this only effects the thread which receives this command.

`end_server` [Command]

Terminate the server program `gwirdsif`. In normal use, the server runs as a *dæmon process*, i.e., it could theoretically run "forever" and never terminate. In addition, it is a *multithreaded process* which may be serving multiple clients at the same time. In normal use, it would therefore be very undesirable if a client were able to terminate the server as a whole, including threads in which it's communicating with other clients.

This command is therefore normally disabled and must be enabled by invoking `gwirdsif` with the '`--end-server-enable`' option. See Section 6.2.8 [Debugging options], page 68. It is needed for *profiling* using `gprof` and `gcov`. See Chapter 30 [Profiling and testing], page 177.

8.9 Raising signals

The commands in this section make it possible for a user to have either the server or the client send a signal to itself.

`signal server` *INTEGER* [Commands]
`signal server` <*string*>

When the server receives one of these commands, it checks if the signal number (*INTEGER*) or name (*STRING*) is valid (see below). If it is, it sends the signal to the main thread of the server program.

If the signal name or number is invalid, an error message is issued and the server continues execution.

`signal client` *INTEGER* [Commands]
`signal client` *STRING*

When the server receives one of these commands, it checks if the signal number (*INTEGER*) or name (*STRING*) is valid (see below). If it is, it creates a "command-only" response which causes the command '`signal INTEGER`' to be returned to the client, which will then send the signal to itself. See Chapter 15 [class Response_Type], page 116.

If the signal name or number is invalid, an error message is issued and the server continues execution.

Signal names and numbers are implementation-dependent, though some signal names and/or numbers are specified by POSIX. See Section "kill invocation" in *GNU Coreutils*.

`gwrdifpk` assumes the existence of the following signals and the association of signal name and number as listed:

1	SIGHUP	Hangup
2	SIGINT	Terminal interrupt
3	SIGQUIT	Terminal quit

4	SIGILL	Illegal Instruction
5	SIGTRAP	Trace/breakpoint trap
6	SIGABRT	Process abort
7	SIGBUS	Access to an undefined portion of a memory object
8	SIGFPE	Erroneous arithmetic operation
9	SIGKILL	Kill (cannot be caught or ignored)
10	SIGUSR1	User-defined signal 1
11	SIGSEGV	Invalid memory reference (segment violation)
12	SIGUSR2	User-defined signal 2
13	SIGPIPE	Write on a pipe with no one to read it
14	SIGALRM	Alarm Clock
15	SIGTERM	Termination
16	SIGSTKFLT	Stack fault on coprocessor (unused)
17	SIGCHLD	Child process terminated, stopped, or continued
18	SIGCONT	Continue executing, if stopped
19	SIGSTOP	Stop executing (cannot be caught or ignored)
20	SIGTSTP	Terminal stop
21	SIGTTIN	Background process attempting read
22	SIGTTOU	Background process attempting write
23	SIGURG	High bandwidth data is available at a socket
24	SIGXCPU	CPU time limit exceeded
25	SIGXFSZ	File size limit exceeded
26	SIGVTALRM	Virtual timer expired
27	SIGPROF	Profiling timer expired
28	SIGWINCH	Window resize signal
29	SIGIO	I/O now possible
30	SIGPWR	Power failure
31	SIGSYS	Bad system call
34 . . . 64	SIGRTMIN . . . SIGRTMAX	Real-time signals, minimum . . . maximum

`SIGRTMIN` and `SIGRTMAX` define the range of available *real-time signals*. Real-time signals > `SIGRTMIN` and < `SIGRTMAX` should be specified as `SIGRTMIN+`*x* where $x = 1 \ldots 15$ for signal numbers $35 \ldots 49$ or `SIGRTMAX-`*y* where $y = 1 \ldots 14$ for signal numbers $50 \ldots 63$. **Please Note:** The specific numbers are implementation dependent, as are the number of real-time signals available (POSIX mandates at least 8).

8.10 TANs

`send tan list` [Command]

This command is currently non-functional. It succeeds on the server-side, but no response is sent to the client. It will only be needed if *TANs* (i.e., *transaction authentication numbers*) are used for authentication/authorization.

8.11 Cryptographic operations

8.11.1 GPG key pairs

`store public_key` [Command]

Send the user's public key to the server and store it in the server-side `gwirdsif.GPG_Key_Pairs` database table. It must already have been stored in the `gwirdcli.GPG_Key_Pairs` database table. As of 2014.01.23., this must have been done "by hand", since no means to do this "automatically" has been implemented yet. See Section 28.14 [GPG_Key_Pair database tables (gwirdsif)], page 172.

8.11.2 Checksums

`checksum` <*path*> [*OPTION . . .*] [Command]

Retrieve a *checksum* for the iRODS object *path* from the server. If a checksum of the specified type (MD5 by default) does not exist, create one.

Options:

`md5`
`sha1`
`sha224`
`sha256`
`sha384`
`sha512` MD5 is the default type of checksum. The options `sha1` through `sha512` may be used to create other types of checksums.

Only one of these options should be used in a single call to the `checksum` command! If more than one is used, the checksum with the largest number of bits takes precedence. That is, SHA512 has the highest precedence and MD5 the lowest.

It is not currently possible to generate multiple checksums using a single call to this command. It would be possible to implement this, but at the present time the author doesn't consider this to be useful feature.

`no-handle`
By default, if a handle value containing the specified type of checksum for the iRODS object *path* does not exist, it is created. This option suppresses creation of the handle value.

`check` By default, if a handle value containing the specified type of checksum for the iRODS object *path* exists, the checksum is simply extracted from the handle value and sent to the client without checking it. If this option is used, `gwirdsif` calls the corresponding checksum function to ensure that the checksum is correct. If it is not, it sends a warning message to the client and tries to update the handle value with the correct checksum.

`verify checksum` <*path*> <*checksum*> [*OPTIONS . . .*] [Command]

Verify checksum *checksum* for iRODS object *path*.

Options:

`md5`
`sha1`
`sha224`
`sha256`
`sha384`
`sha512` MD5 is the default type of checksum. The options `sha1` through `sha512` may be used to verifyother types of checksums.

Only one of these options should be used in a single call to the `checksum` command! If more than one is used, the checksum with the largest number of bits takes precedence. That is, SHA512 has the highest precedence and MD5 the lowest.

It is not currently possible to verify multiple checksums using a single call to this command. It would be possible to implement this, but at the present time the author doesn't consider this to be useful feature.

`no-handle`
This option has no effect. It exists because this command and the `checksum` command (see above) use the same options.

`check` By default, if a handle value containing the specified type of checksum for the iRODS object *path* exists, the checksum is simply extracted from the handle value and compared with the *checksum* sent by the client without checking it. If this option is used, `gwirdsif` calls the corresponding checksum function to ensure that the checksum is correct. If it is not, it sends a warning message to the client and tries to update the handle value with the correct checksum.

8.12 Pull requests

`register pull` [*OPTION* ...] [Command]
Register pull request. See Chapter 7 [Pull archiving], page 72, and Section 28.15 [Pull Request database table (gwirdsif)], page 172.

8.13 Other user commands

`end` [Command]
Tell the server that the client is finished. Any input following this command is ignored by the server.

9 User commands for controlling the client

The commands described in the previous chapter, Chapter 8 [User commands], page 73, are entered on the client-side, but passed by the client directly to the server without being processed by the client in any way. The commands described in this chapter, on the other hand, are processed by the client, but separately from the commands sent to the client by the server. That is, whereas `gwirdcli` processes the commands sent from the server by means of the scanner/parser pair `zzlex/zzparse`, the client-side commands described in this chapter are processed by means of the scanner/parser pair `xxlex/xxparse`. See Section 26.6 [Scanning and parsing], page 152.

9.1 Pull responses

`add pull path` [*OPTION ...*] `<local path> <remote path>` [...] [Commands]
`add pull paths` [*OPTION ...*] `<local path> <remote path>`
[`<local path> <remote path>` ...]

Add one or more *pull paths* to a pull response. If a corresponding pull response doesn't already exist, create it. These commands are synonymous; '`add pull paths`' can be used for a single pair of paths, while '`add pull path`' can be used for multiple pairs.

These commands insert rows into the client-side database tables '`gwirdcli.Pull_Responses`' and '`gwirdcli.Pull_Paths`'.

Options:

`user_id` *INTEGER*
`username` *STRING*
`distinguished_name` *STRING*
: !! TODO: Check how these are used! LDF 2014.02.27. **Please note**: Creating a pull response or adding a pull path for a different user requires the '`pull_response_group`' or '`pull_response_all`' privilege. See Section 29.4 [Privileges_Gwirdcli database table (gwirdcli)], page 174.

`server_ip_address` *STRING*
: Required. The IP address of the server which will be sending the *pull request*.

`server_hostname` *STRING*
: Optional. The hostname of the server which will be sending the *pull request*.

`client_ip_address` *STRING*
: The IP address of the pull client. The default is the IP address of the computer on which `gwirdcli` is running.

`client_hostname` *STRING*
: The hostname of the pull client. The default is the hostname of the computer on which `gwirdcli` is running.

`force` Add the pull paths, even if corresponding ones exist.

`interval` *INTEGER*
: Set the pull interval. Default: 86400, i.e., one day in seconds.

See also Chapter 7 [Pull archiving], page 72, and Section 29.7 [Pull Response database tables (gwirdcli)], page 175.

10 User and Group Management

10.1 Privileges

11 Using gwirdsif

11.1 Socket and log directories

11.2 Purging

See also Section 6.2.10 [Purging options], page 69.

11.3 Signal handling

12 Using gwirdcli

12.1 Purging

See also Section 6.2.10 [Purging options], page 69.

13 Global constants and variables

The global constants and variables are declared within the *namespace* `gwrdifpk`.

13.1 Global constants

The global constants are declared and initialized in '`glblcnst.web`'. Unlike the global variables (see below), they can be included in the '`gwrdifpk`' library, because they are constant, i.e., their values never change. Their values can therefore be shared by any number of running processes, with no danger of them being overwritten.

All of the global constants listed below are declared `extern const`.

`int DEFAULT_PORT_NUM_ANON` [Global constants]
`string DEFAULT_PORT_STR_ANON`
Values: `5556` and `"5556"`, respectively.

`string DEFAULT_PORT_STR_X_509` [Global constants]
`int DEFAULT_PORT_NUM_X_509`
Values: 5557 and `"5557"`, respectively.

`string DEFAULT_SOCKET_DIRECTORY` [Global constant]
Value: `"/tmp"`

`size_t MYSQL_PASSWORD_LENGTH` [Global constant]
Value: `256`

`string DEFAULT_MYSQL_PASSWORD_FILENAME` [Global constant]
Value: `"mysql_password.gpg.asc"`

`string DEFAULT_LISTEN_CLIENT_PORT_STR` [Global constant]
Value `"5558"`

`int DEFAULT_LISTEN_CLIENT_PORT` [Global constant]
Value: `5558`

`int DEFAULT_PULL_REQUEST_INTERVAL` [Global constant]
Value: 259200, i.e., three days in seconds

13.1.1 Response and error codes

```
extern const int GW_SUCCESS                                  [Global constants]
extern const int GW_ERROR
extern const int GW_WARNING
extern const int GW_SERVER_SIDE_DATABASE_ERROR
extern const int GW_NO_PRIVILEGE_ERROR
extern const int GW_HANDLE_NOT_FOUND
extern const int GW_HANDLE_NOT_MARKED_FOR_DELETION
extern const int GW_HANDLE_ALREADY_MARKED_FOR_DELETION
extern const int GW_HANDLE_VALUE_ERROR
extern const int GW_INVALID_HANDLE_VALUE_SPECIFIER
```

```
extern const int GW_HANDLE_VALUE_NOT_FOUND
extern const int GW_LAST_HANDLE_VALUE
extern const int GW_HS_ADMIN_HANDLE_VALUE
extern const int GW_LAST_HS_ADMIN_HANDLE_VALUE
extern const int GW_HANDLE_VALUE_NOT_MARKED_FOR_DELETION
extern const int GW_HANDLE_VALUE_ALREADY_MARKED_FOR_DELETION
extern const int GW_IRODS_OBJECT_NOT_FOUND
extern const int GW_IRODS_OBJECT_NOT_MARKED_FOR_DELETION
extern const int GW_IRODS_OBJECT_ALREADY_MARKED_FOR_DELETION
extern const int GW_DC_METADATA_NOT_FOUND
extern const int GW_DC_METADATA_NOT_MARKED_FOR_DELETION
extern const int GW_DC_METADATA_ALREADY_MARKED_FOR_DELETION
```

These constants are declared in '`gwrdifpk-1.0/src/rspercds.web`'. They are numbered consecutively, whereby `GW_SUCCESS` = 0 and `GW_ERROR` = 1. Otherwise, the specific values are not significant, as long as they are distinct and > 1. They are sent from the server to the client and used in the client-side parser function `zzparse`. See Section 26.6 [Scanning and parsing], page 152. The function `gwstrerror` can be used to output a human-readable message for a given response or error code. See Section 26.15 [Other functions], page 156.

13.2 Global variables

The global variables are declared and initialized in '`glblvrbl.web`'. Unlike the global constants (see above), they cannot be included in a library, because if they are shared among several processes, each variable would contain the value stored in it by the last process that assigned to it. Therefore, every program that uses the global variables links with the *object file* (or file of object code) '`glblvrbl.o`' so that each instance of each program has its own copy of all of the global variables.

Another way of solving this problem would be to put all of the global variables into a class and declare an instance of this class in each program. There are advantages and disadvantages to both of these approaches.

Many of the global variables described below are set by the *command-line* options specified by the user when invoking `gwirdsif` or `gwirdcli`. See Chapter 6 [Invoking gwirdsif/gwirdcli], page 63.

```
unsigned int purge_server_logs_thread_ctr                [Global variables]
unsigned int purge_database_thread_ctr
unsigned int purge_irods_archive_thread_ctr
unsigned int purge_dc_metadata_thread_ctr
```

```
unsigned int listen_local_thread_ctr                     [Global variables]
unsigned int listen_remote_anon_thread_ctr
unsigned int listen_remote_X_509_thread_ctr
```

```
unsigned int pull_request_thread_ctr                     [Global variable]
```

```
unsigned int save_global_thread_ctr                      [Global variable]
```

Initialized to 7, i.e., the value of `listen_remote_X_509_thread_ctr + 1`.

`unsigned int global_thread_ctr` [Global variable]

Initialized to 6, i.e., the value of `listen_remote_X_509_thread_ctr`. The first thread created other than the ones named above ("purge server", "purge database" ... "listen_remote_X_509_thread_ctr" will be thread number 7.

`bool global_thread_ctr_wrapped_around` [Global variable]

Initialized to `false`. Currently only used by `gwirdsif`, because `gwirdcli` doesn't use threads. (This may change in the future.)

This variable keeps track of whether `global_thread_ctr` has wrapped around. Since the maximum value of an `unsigned int` is very large (`UINT_MAX` $\equiv 4,294,967,295$ on the author's PC), it is extremely unlikely that `global_thread_ctr` would ever wrap around, even if `gwirdsif` were to run uninterrupted for a very long time. However, it is not impossible and a test procedure could be programmed to make it happen. Otherwise, if `global_thread_ctr_wrapped_around` were ever to be set to `true`, it would be an indication that something has probably gone wrong.

`pthread_mutex_t thread_ctr_id_map_mutex` [Global variable]

`pthread_t purge_server_logs_thread_id` [Global variables]
`pthread_t purge_database_thread_id`
`pthread_t purge_irods_archive_thread_id`
`pthread_t purge_dc_metadata_thread_id`

`pthread_mutex_t thread_ctr_mutex` [Global variables]
`pthread_mutex_t cerr_mutex`
`pthread_mutex_t cout_mutex`
`pthread_mutex_t log_strm_mutex`
`pthread_mutex_t err_log_strm_mutex`
`pthread_mutex_t sql_mutex`
`pthread_mutex_t sql_lock_tables_mutex`
`pthread_mutex_t session_data_mutex`
`pthread_mutex_t gpg_passphrase_fifo_mutex`

`map<unsigned int, pthread_t> thread_ctr_id_map` [Global variables]
`map<pthread_t, unsigned int> thread_id_ctr_map`

`ofstream output_file_strm` [Global variables]
`ofstream log_strm`
`ofstream err_log_strm`

`string server_ip_address` [Global variable]

`string input_filename` [Global variables]
`string output_filename`

`string port_str_anon` [Global variables]
`string port_str_x_509`

`int port_num_anon` [Global variables]
`int port_num_x_509`

`bool save_temp_files` [Global variable]

`map<string, string> dn_fields` [Global variables]
`map<string, string> dn_username_map`

`string DEFAULT_CERT_FILENAME` [Global variables]
`string DEFAULT_KEY_FILENAME`
`string DEFAULT_CA_FILENAME`
`string DEFAULT_CRL_FILENAME`

`string log_filename` [Global variables]
`string err_log_filename`

`int trace_value` [Global variables]
`bool scanner_trace`
`bool parser_trace`

`string gwirdsif_hostname` [Global variables]

`bool icommands` [Global variables]
`bool irods_functions`
`bool jargon_trunk`
`bool jargon_core`
`icommands` is `true` by default and the others are `false`.

`int irods_server_pid` [Global variables]

`string irods_server_dir` [Global variables]

`string config_dir` [Global variables]

`char[64] admin_data` [Global variables]
`unsigned int admin_data_length`

`bool end_server_enabled` [Global variables]
`bool sleep_server_enabled`
`bool sleep_client_enabled`
`bool signal_server_enabled`
`bool signal_client_enabled`

`unsigned int purge_logs_interval` [Global variables]
604800U 1 week in seconds

`unsigned int purge_database_interval` [Global variables]
3600U 1 hour in seconds

`unsigned int purge_database_limit` [Global variables]
172800U 2 days in seconds

`pthread_mutex_t purge_server_database_mutex` [Global variables]
`pthread_cond_t purge_server_database_cond`

`pthread_mutex_t purge_dc_metadata_mutex` [Global variables]
`pthread_cond_t purge_dc_metadata_cond`

`pthread_mutex_t purge_irods_archive_mutex` [Global variables]
`pthread_cond_t purge_irods_archive_cond`

`unsigned int purge_irods_archive_interval` [Global variables]
3600U 1 hour in seconds

`unsigned int purge_irods_archive_limit` [Global variable]
172800U 2 days in seconds

`unsigned int purge_logs_limit` [Global variable]
14U 14 days

`unsigned int purge_dc_metadata_interval` [Global variable]
3600U 1 hour in seconds

`unsigned int purge_dc_metadata_limit` [Global variable]
172800U 2 days in seconds

`string socket_dir` [Global variable]

`string socket_path` [Global variable]

`string log_dir` [Global variable]

`struct sigaction default_sigint_action` [Global variable]

`struct sigaction default_sigterm_action` [Global variable]

`bool terminate_on_end_input` [Global variable]
true Used by gwirdcli only.

`string command_str` [Global variable]

`int global_debug_level` [Global variable]
Default 0.

`unsigned int suppress_prompt` [Global variable]
Default 0U.

`string mysql_username` [Global variable]

`string mysql_password_filename` [Global variable]

`char* mysql_password` [Global variable]
Default 0.

`vector<string> cert_filenames` [Global variable]

`vector<string> key_filenames` [Global variable]

`vector<string> ca_filenames` [Global variable]

`vector<string> crl_filenames` [Global variable]

`string session_id` [Global variable]

`string homedir` [Global variable]

`string gwirdsif_dir` [Global variable]

`string gpg_homedir` [Global variable]

`char* gpg_passphrase` [Global variable]
Default 0.

`size_t gpg_passphrase_length` [Global variable]
Default 0.

`char* gpg_key_id` [Global variable]
Default 0.

`int gpg_passphrase_fifo_fd` [Global variable]
Default 0.

`string gpg_passphrase_fifo_name` [Global variable]

`vector<string> session_id_vector` [Global variable]

`bool is_gwrdwbap` [Global variable]
Initialized to `false`.

`bool is_gwirdcli` [Global variable]
Initialized to `false`.

`bool is_gwirdsif` [Global variable]
Initialized to `false`.

`bool is_gwirdpcl` [Global variable]
Initialized to `false`.

`string remote_hostname` [Global variable]

`bool remote_connection` [Global variable]
Initialized to `false`.

`bool anonymous` [Global variable]
Initialized to `false`.

`bool standalone_handle` [Global variable]
Initialized to `true`. This is for testing purposes! For production use, this variable should be initialized to `false`.

`multimap<string, int> signal_number_map` [Global variable]

`map<int, string> signal_name_map` [Global variable]

`map<int, string> gw_code_map` [Global variable]
Declared in '`gwrdifpk-1.0/src/rspercds.web`'. See Section 13.1.1 [Response and error codes], page 101. It is initialized (filled) by the function `init_gw_code_map` and used by the function `gwstrerror`. See Section 26.15 [Other functions], page 156.

`int listen_client_port` [Global variable]

`string listen_client_port_str` [Global variable]

`int pull_request_interval` [Global variable]

`bool listen_client` [Global variable]

14 class Scan_Parse_Parameter_Type

`class Scan_Parse_Parameter_Type` is a very important data type within `gwrdifpk`: It represents the "state" of a session. That is, on the client-side, it represents the state of a run of the program `gwirdcli`, whereas on the server-side, it represents the state of a thread in which the server program `gwirdsif` is communicating with an instance of the client.

The reason this class is named "`Scan_Parse_Parameter_Type`" is because an object of this type is passed as a parameter to the scanner and parser functions `yylex` and `yyparse` on the server-side and `zzlex` and `zzparse` on the client-side. See Section 26.6 [Scanning and parsing], page 152.

`class Scan_Parse_Parameter_Type` contains many variables to represent the state of the session and many of the functions called in the course of a session are member functions of this class. The connections to the various MySQL databases are also managed by means of data members and member functions belonging to this class.

The following functions are `friends` of `class Scan_Parse_Parameter_Type`:

`client_func`
: See Section 26.1 [Main (and similar) functions], page 151.

`client_sending_file_rule_func`
: See Section 26.7 [Parser rule functions], page 153.

`connect_func`
: See Section 26.3 [Listen and connect functions], page 151.

`create_databases`
: Defined in '`stpcrdbs.web`' and called in the `main` function of `setupdbs` ('`setupdbs.web`'). See Section 2.5 [Database setup], page 7, and Section 32.5 [Set up databases (setupdbs)], page 185.

`distinguished_name_rule_func`
`exchange_data_with_client`
`exchange_data_with_server`
`get_avus_from_irods_system`
`get_user_info_func`
`listen_local`
: See Section 26.3 [Listen and connect functions], page 151.

`listen_remote_X_509`
: See Section 26.3 [Listen and connect functions], page 151.

`listen_remote_anon`
: See Section 26.3 [Listen and connect functions], page 151.

`main` See Section 26.1 [Main (and similar) functions], page 151.

`parse_post_data`
`pull_response`
`pull_client_func`
`yylex`

`yyparse`

`Irods_Object_Type::get_avus_from_irods_system`

14.1 Data Members

14.1.1 Static constants

Static `unsigned int` constants in `Scan_Parse_Parameter_Type`:

Name	Value
NULL_AUTH_TYPE	0
LOCAL_NULL_AUTH_TYPE	1
X_509_AUTH_TYPE	2
ANON_AUTH_TYPE	3

Privileges are represented as bit positions in `unsigned int` values used as bit fields. These constants are numbered in powers of 2 from $1^0 = 1$ to $2^{12} = 2048$. On the author's PC, 32 privileges are allowed, since `unsigned int` values are of length 4 bytes = 32 bits.

If more privileges are needed at a later date, `unsigned long int` values will have to be used instead.

Name	Value
SUPERUSER_PRIVILEGE	1
DELEGATE_PRIVILEGE	2
DELETE_HANDLES_PRIVILEGE	4
DELETE_HANDLE_VALUES_PRIVILEGE	8
DELETE_HS_ADMIN_HANDLE_VALUES_PRIVILEGE	16
DELETE_LAST_HS_ADMIN_HANDLE_VALUE_PRIVILEGE	32
UNDELETE_HANDLE_VALUES_PRIVILEGE	64
SHOW_USER_INFO_PRIVILEGE	128
SHOW_GROUPS_PRIVILEGE	256
SHOW_CERTIFICATES_PRIVILEGE	512
SHOW_DISTINGUISHED_NAMES_PRIVILEGE	1024
SHOW_PRIVILEGES_PRIVILEGE	2048

14.1.2 Static variables

The following variables use the data type `Scan_Parse_Parameter_Type::func_ptr` which is defined as:

```
typedef int (Scan_Parse_Parameter_Type::*func_ptr)(Response_Type &)
```

That is, `Scan_Parse_Parameter_Type::func_ptr` is a pointer to a function taking a reference to `Response_Type` as its argument and returning `int`. See Section 14.2.12 [Server action functions], page 114, and Section 14.2.13 [Client action functions], page 115.

```
map<unsigned int, func_ptr> server_action_map          [Static variables]
map<unsigned int, func_ptr> client_action_map
```

```
map<unsigned int, string> server_action_name_map       [Static variables]
map<unsigned int, string> client_action_name_map
```

14.1.3 Variables

`int sock` [Variable]

`gnutls_session_t session` [Variable]

`bool remote_connection` [Variable]

`bool anonymous` [Variable]

`unsigned int connection_type` [Variable]

`bool PARSER_DEBUG` [Variable]

`MYSQL* mysql_ptr` [Variable]

`Distinguished_Name_Type distinguished_name` [Variable]

`int user_id` [Variable]

`string username` [Variable]

`vector<Group_Type> group_vector` [Variable]

`unsigned int privileges` [Variable]

On the author's PC, this allows the definition of 32 privileges. See Section 14.1.1 [Static constants], page 108, above.

`string irods_auth_filename` [Variable]

`string irods_env_filename` [Variable]

`string irods_password_encrypted` [Variable]

`string irods_password_encrypted_timestamp` [Variable]

`string irods_homedir` [Variable]

`string irods_current_dir` [Variable]

`string irods_zone` [Variable]

`string irods_default_resource` [Variable]

`int thread_ctr` [Variable]

`time_t expires` [Variable]

`string data_filename` [Variable]

`char data_buffer[BUFFER_SIZE]` [Variable]

`string input_commands` [Variable]

`deque<Response_Type> response_deque` [Variable]

`deque<Response_Type> delayed_response_deque` [Variable]

`bool pending_operations_flag` [Variable]

`deque<Response_Type>::iterator pending_operations_iter` [Variable]

`map<unsigned int, Response_Type> response_map` [Variable]

pthread_mutex_t response_map_mutex [Variable]

bool client_finished [Variable]

bool server_finished [Variable]

vector<string> filename_vector [Variable]

unsigned int default_handle_prefix_id [Variable]

string default_handle_prefix [Variable]

unsigned int default_institute_id [Variable]

string default_institute_name [Variable]

string pid_str [Variable]

string pid_prefix_str [Variable]

string pid_suffix_str [Variable]

string pid_institute_str [Variable]

unsigned long *delay_value* [Variable]

Handle_Value_Triple hvt [Variable]

map<string, int> *user_id_map* [Variable]

map<int, User_Info_Type> user_info_map [Variable]

Handle_Value_Type handle_value [Variable]

User_Info_Type* user_info_ptr [Variable]

Irods_Object_Type* irods_object [Variable]

vector<Irods_Object_Type> irods_object_vector [Variable]

vector<string> temp_file_vector [Variable]

vector<string> string_vector [Variable]

vector<int> int_vector [Variable]

unsigned int errors_occurred [Variable]

unsigned int warnings_occurred [Variable]

int thread_cancel_state [Variable]

X509_Cert_Type user_cert [Variable]

See Chapter 20 [X.509 Certificate Types], page 134.

X509_Cert_Type server_cert [Variable]

X509_Cert_Type ca_cert [Variable]

X509_Cert_Type* cert_ptr [Variable]

string public_key_id [Variable]

string temp_gpg_key_fingerprint [Variable]

unsigned int gpg_key_pair_id [Variable]

string gpg_key_fingerprint [Variable]

14.2 Member Functions

`Scan_Parse_Parameter_Type` has so many member functions that the definitions are spread over several files: '`scprpmtp.web`', '`spptfnc1.web`' and '`spptfnc2.web`' contain "normal" member functions, while the files '`srvractn.web`' and '`clntactn.web`' contain the definitions for "server action" and "client action" functions, respectively. See Section 14.2.12 [Server action functions], page 114, and Section 14.2.13 [Client action functions], page 115, below.

14.2.1 Constructor and initialization functions

`void Scan_Parse_Parameter_Type (void)` [Default constructor]

`Scan_Parse_Parameter_Type` only has this one constructor.

`int initialize_maps (void)` [Static function]

14.2.2 User and group administration

```
int get_user (int curr_user_id = 0,                          [Function]
        const char *dn = 0,
        string curr_username = "",
        User_Info_Type *user_info = 0,
        bool set_user = false)
```

```
int set_user_info (User_Info_Type &user_info)                [const function]
```

```
int get_database_username (void)                             [Function]
```

```
int get_privileges (int curr_user_id = 0,                    [Function]
        unsigned int *privs = 0)
```

```
int show_privileges (unsigned int privileges,                [Static function]
        ostream *strm = 0,
        bool verbose = false)
```

14.2.3 X.509 certificates

```
int show_certificates (Response_Type &response,              [Function]
        char *buffer,
        size_t buffer_size,
        string &filename)
```

14.2.4 Communication

```
int get_input (void)                                         [Function]
```

```
int send_to_peer (char **buffer_ptr,                         [Function]
        unsigned int char_ctr = 0,
        string filename = "")
```

`int send_to_peer (const Response_Type &response)` [Function]

`int receive_file (string remote_filename = "",` [Function]
`string local_filename = "",`
`bool overwrite = false,`
`string *new_local_filename_ptr = 0,`
`string *temp_filename_ptr = 0)`

14.2.5 iRODS

`int cd (string dir,` [Function]
`char *buffer_ptr,`
`unsigned int buff_size)`

`int mkdir (Response_Type &response,` [Function]
`char *buffer_ptr,`
`size_t buffer_size)`

`int ls (char *buffer_ptr,` [Function]
`unsigned int buff_size,`
`string *filename,`
`Response_Type *response,`
`string filename_1 = "",`
`bool do_response = true)`

`int mv (Response_Type &response,` [Function]
`string thread_str = "")`

`int pwd (char *buffer_ptr,` [Function]
`unsigned int buff_size,`
`string args = "")`

`int put (Response_Type &response)` [Function]

`int get (Response_Type &response,` [Function]
`string thread_str = "")`

`int mark_irods_objects_for_deletion (Response_Type` [Function]
`&response,`
`char *buffer_ptr,`
`size_t buffer_size)`

`int undelete_files (Response_Type &response,` [Function]
`string thread_str = "")`

14.2.6 Handles

`int fetch_handle_from_database (unsigned long int handle_id,` [Function]
`Handle_Type &handle,`
`string type = "")`

`int fetch_handle_from_database (string handle_str,` [Function]
`Handle_Type &handle,`
`string type = "")`

```
int fetch_handles_from_database (vector<unsigned long int>          [Function]
        &handle_id_vector,
        vector<Handle_Type> &handle_vector,
        string type = "")
int get_handle (string s,                                           [Function]
        unsigned int flags,
        unsigned int options = 0U,
        string filename_1 = "")
```

14.2.7 Dublin Core metadata

```
int add_metadata (Response_Type &response)                          [Function]
int store_dc_metadata (const Response_Type &response,               [Function]
        Handle_Type &irods_object_handle,
        Handle_Type &dc_metadata_handle,
        bool force,
        string &irod_object_path,
        unsigned long int dc_metadata_id,
        unsigned long int irods_object_ref_id = 0UL,
        string thread_str = "")
int parse_metadata (vector<Dublin_Core_Metadata_Type>               [Function]
        &dc_metadata_vector,
        Response_Type &response)
int get_metadata (string filename,                                  [Function]
        unsigned int flags,
        int *ctr = 0,
        unsigned int options = 0,
        char *buffer_ptr = 0,
        size_t buffer_size = 0,
        bool do_output = true,
        bool do_irods_user_metadata = true,
        map<unsigned long int, Dublin_Core_Metadata_Type> *
        dc_metadata_type_map_ptr = 0)
```

14.2.8 Database

```
int submit_mysql_query (string query, MYSQL_RES *&result,           [Function]
        unsigned int *row_ctr,
        unsigned int *field_ctr,
        long *affected_rows = 0)
int submit_mysql_queries (vector<string> &query_vector,             [Function]
        MYSQL_RES **result_array,
        vector<unsigned int *> &row_ctr_vector,
        vector<unsigned int *> &field_ctr_vector,
        vector<long int *> &affected_rows_vector,
        bool continue_on_error = false)
```

```
int get_highest_value (MYSQL *mysql_ptr,                    [Static function]
        string table,
        string column,
        unsigned long int& val,
        bool incr = false)
```

14.2.9 Cryptographic operations

```
int generate_checksum (Response_Type &response,             [Function]
        bool verify_only = false,
        string thread_str = "")
```

```
int store_public_key (string uid,                           [Function]
        string fingerprint,
        string public_key,
        unsigned int options,
        string thread_str = "")
```

14.2.10 TANs

```
int send_tan_list (void)                                    [Function]
```

14.2.11 Other functions

```
void show (string s = "Scan_Parse_Parameter_Type:",         [Function]
        stringstream *strm = 0)
```

```
int set_expires (time_t seconds,                            [Function]
        stringstream *out_strm = 0)
time_t get_expires (char *str = 0,                          [Function]
        size_t str_size = 0,
        stringstream *out_strm = 0)
```

These functions are only called in the web application `gwrdwbap`. See Chapter 31 [Web application gwrdwbap], page 178.

14.2.12 Server action functions

```
int server_action_delete_handle_value (                     [Functions]
        Response_Type &response)
int server_action_add_handle_value (Response_Type &response)
int server_action_cd (Response_Type &response)
int server_action_command_only (Response_Type &response)
int server_action_create_handle (Response_Type &response)
int server_action_delete_handle (Response_Type &response)
int server_action_end_server (Response_Type &response)
int server_action_get (Response_Type &response)
int server_action_get_handle (Response_Type &response)
int server_action_get_metadata (Response_Type &response)
int server_action_get_user_info (Response_Type &response)
int server_action_ls (Response_Type &response)
```

```
int server_action_mv (Response_Type &response)
int server_action_mark_irods_objects_for_deletion
        (Response_Type &response)
int server_action_mkdir (Response_Type &response)
int server_action_process_pending (Response_Type &response)
int server_action_pwd (Response_Type &response)
int server_action_receive_metadata_file (Response_Type &response)
int server_action_receive_put_file (Response_Type &response)
int server_action_send_file (Response_Type &response)
int server_action_send_handle (Response_Type &response)
int server_action_send_metadata (Response_Type &response)
int server_action_send_tan_list (Response_Type &response)
int server_action_show_certificate (Response_Type &response)
int server_action_sleep (Response_Type &response)
int server_action_undelete_file (Response_Type &response)
int server_action_undelete_handle (Response_Type &response)
int server_action_undelete_handle_value (Response_Type &response)
int server_action_delete_metadata (Response_Type &response)
int server_action_undelete_metadata (Response_Type &response)
int server_action_show_metadata (Response_Type &response)
int server_action_generate_checksum (Response_Type &response)
int server_action_verify_checksum (Response_Type &response)
int server_action_store_public_key (Response_Type &response)
int server_action_unknown (Response_Type &response)
```

14.2.13 Client action functions

[Functions]

```
int client_action_command_only (Response_Type &response)
int client_action_send_file (Response_Type &response)
int client_action_unknown (Response_Type &response)
int client_action_send_public_key (Response_Type &response)
```

15 class Response_Type

`class Response_Type` is used to store information needed for the communication between the *peers* `gwirdsif` and `gwirdcli`. Objects of this type are created in *parser actions*, i.e., in the functions `yyparse` (on the server-side) and `zzparse` (on the client-side), and in functions called from the parser actions. See Section 26.6 [Scanning and parsing], page 152. They are then pushed onto the `deque<Response_Type> response_deque` data member belonging to the `Scan_Parse_Parameter_Type` object that was passed as a parameter to the parser function (i.e., `yyparse` or `zzparse`). See Section 14.1.3 [Variables (Scan_Parse_Parameter_Type)], page 109. In the course of processing, they are *popped* from the `deque` in the function `exchange_data_with_client` (server-side) or `exchange_data_with_server` (client-side) and passed as parameters to the appropriate *server action* or *client action* function. See Section 14.2.12 [Server action functions], page 114, and Section 14.2.13 [Client action functions], page 115.

The following classes and functions are `friends` of `class Response_Type`:

```
class Handle_Value_Type;
class Irods_Object_Type
class Scan_Parse_Parameter_Type
class Dublin_Core_Metadata_Type
client_sending_file_rule_func
distinguished_name_rule_func
exchange_data_with_client
exchange_data_with_server
int get_user_info_func
yyparse
```

15.1 Data Members

[Private variables]

```
unsigned int type
string local_filename
string remote_filename
string temporary_filename
string dirname
string command
string flags
unsigned int options
unsigned int metadata_options
unsigned int pid_options
string pid_str
string pid_prefix_str
string pid_suffix_str
string pid_institute_str
string gpg_key_fingerprint
Handle_Type* handle
Handle_Value_Triple hvt
int int_val
```

```
string string_val
bool no_delay
unsigned long int delay_value
vector<string> string_vector
vector<int> int_vector
```

`map<unsigned int, string> typename_map` [static public variable]

Public static `unsigned int` constants in `Response_Type`:

Name	Value
NULL_RESPONSE_TYPE	0
COMMAND_ONLY_TYPE	1
SEND_FILE_TYPE	2
RECEIVE_PUT_FILE_TYPE	3
RECEIVE_METADATA_FILE_TYPE	4
SEND_HANDLE_TYPE	5
LS_TYPE	6
MV_TYPE	7
PWD_TYPE	8
CD_TYPE	9
MKDIR_TYPE	10
UNDELETE_FILE_TYPE	11
MARK_IRODS_OBJECTS_FOR_DELETION_TYPE	12
GET_TYPE	13
SEND_METADATA_TYPE	14
END_SERVER_TYPE	15
SLEEP_TYPE	16
SHOW_CERTIFICATE_TYPE	17
GET_METADATA_TYPE	18
GET_HANDLE_TYPE	19
SEND_TAN_LIST_TYPE	20
PROCESS_PENDING_TYPE	21
GET_USER_INFO_TYPE	22
CREATE_HANDLE_TYPE	23
ADD_HANDLE_VALUE_TYPE	24
DELETE_HANDLE_TYPE	25
UNDELETE_HANDLE_TYPE	26
DELETE_HANDLE_VALUE_TYPE	27
UNDELETE_HANDLE_VALUE_TYPE	28
DELETE_METADATA_TYPE	29
UNDELETE_METADATA_TYPE	30
FETCH_DC_METADATA_TYPE	31
GENERATE_CHECKSUM_TYPE	32
VERIFY_CHECKSUM_TYPE	33
STORE_PUBLIC_KEY_TYPE	34
SEND_PUBLIC_KEY_TYPE	35
MAX_RESPONSE_TYPE	35

`MAX_RESPONSE_TYPE` should always have the highest value assigned to another |Response_Type| constant.

15.2 Member Functions

`void Response_Type (void)` [Default constructor]

`void Response_Type (const Response_Type &r)` [Copy constructor]

`void ~Response_Type (void)` [Destructor]

```
const Response_Type& operator= (
        const Response_Type &r)
```
[Assignment operator]

`int initialize_maps (void)` [static function]

`void clear (void)` [Function]

`int show (string s = "")` [`const` function]

16 class User_Info_Type

`class User_Info_Type` represents user information within `gwrdifpk`. See also Section 28.1 [Users (database table)], page 160, and Section 28.2 [User_Info (database view)], page 160.

`class Scan_Parse_Parameter_Type` and the functions `client_func`, `yyparse distinguished_name_rule_func` and `get_user_info_func` are `friends` of `class User_Info_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, Section 26.6 [Scanning and parsing], page 152, and Section 26.7 [Parser rule functions], page 153.

16.1 Data members

```
int user_id                                                    [Private variables]
string username
Distinguished_Name_Type distinguished_name
unsigned int privileges
string irods_password_encrypted
string irods_password_encrypted_timestamp
string irods_current_dir
string irods_homedir
string irods_zone
string irods_default_resource
string irods_env_filename
string irods_auth_filename
unsigned int default_handle_prefix_id
string default_handle_prefix
unsigned int default_institute_id
string default_institute_name
string handle_username
string handle_password_encrypted
X509_Cert_Type certificate
unsigned int gpg_key_pair_id
string gpg_key_fingerprint
```

```
string public_key_id                                           [Public variable]
```

16.2 Member functions

```
void User_Info_Type (void)                                     [Default constructor]
```

```
void clear (void)                                              [Function]
```

```
int get_user_id (void)                                         [Inline functions]
string get_irods_env_filename (void)
string get_irods_auth_filename (void)
```

```
void show (string s = "",                                      [const function]
        stringstream *strm = 0,
        bool brief = false)
```

```
int get_user_info_from_database (MYSQL *mysql_ptr,          [Function]
        string distinguished_name_str,
        string username_str = "",
        string thread_str = "")
```

16.3 Global non-member variables

```
map<int, User_Info_Type> global_user_info_map          [Global variables]
pthread_mutex_t global_user_info_map_mutex
```

17 class Group_Type

`class Group_Type` represents *groups* within `gwrdifpk`. As of 2013.10.28., *group management* has only been implemented in a rudimentary way. See Chapter 10 [User and Group Management], page 98, and Section 28.4 [Groups database tables and views], page 162.

17.1 Data members

`class Scan_Parse_Parameter_Type` and the function `yyparse` are `friends` of `Group_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, and Section 26.6 [Scanning and parsing], page 152.

```
int group_id                                                    [Private variables]
string group_name
map<int, pair<string, unsigned int> > member_id_map
int creator_id
string creator_username
time_t created
```

```
unsigned int NULL_PRIVILEGE                                     [Public static constants]
unsigned int ADD_USER_PRIVILEGE
unsigned int DELETE_USER_PRIVILEGE
unsigned int DELETE_GROUP_PRIVILEGE
```

Numbered `0U`, `1U`, `2U`, `4U`, i.e., consecutive powers of two. For use in *bit fields*.

17.2 Member functions

`void Group_Type (void)` [Default constructor]

`void ~Group_Type (void)` [Destructor]

`int set (MYSQL_ROW &curr_row, string thread_str = "")` [Function]

`int clear (void)` [Function]

`int show (string s = "", stringstream *strm = 0)` [const function]

`int get_from_database (MYSQL *mysql_ptr, string ggroup_name)` [Function]

`int write_to_database (MYSQL * mysql_ptr)` [Function]

```
int get_all_groups (MYSQL *mysql_ptr,                           [static function]
        vector<Group_Type> &group_vector,
        int thread_ctr = 0)
```

18 iRODS Types

`class Irods_Object_Type` and `class Irods_AVU_Type` represent iRODS objects and *AVUs* (i.e., *Attribute-Value-Unit* triples) within the programs belonging to the `gwrdifpk` package. The data members correspond to the fields of the tables `Irods_Objects` and `Irods_AVUs` in the `gwirdsif` database. See Section 28.10.1 [Irods_Objects database table], page 165, and Section 28.10.2 [Irods_AVUs database table], page 166.

Please note that *contents* of the actual iRODS objects are *not* stored, neither in objects of `class Irods_Object_Type` nor in rows in the `gwirdsif.Irods_Objects` database table! There would be no benefit in doing so, since the contents may be retrieved from the iRODS system at any time. In addition, iRODS objects may contain substantial amounts of data, which would affect performance.

The "contents" of the AVUs, on the other hand, *are* stored, both in objects of type `class Irods_AVU_Type` and in rows in the `gwirdsif.Irods_AVUs` database table. Since they are intended to consist of relatively short strings, there is a performance benefit in keeping them available instead of having to retrieve them from the iRODS system when needed.

18.1 class Irods_Object_Type

`class Scan_Parse_Parameter_Type` and the functions `zzparse` and `purge_irods_archive` are `friends` of `class Irods_Object_Type`.

18.1.1 Data Members

[Private variables]

```
unsigned long int id
int user_id
int irods_server_id
string path
bool marked_for_deletion_from_archive
bool marked_for_deletion_from_gwirdsif_db
bool deleted_from_archive
bool deleted_from_gwirdsif_db
time_t delete_from_archive_timestamp
time_t delete_from_gwirdsif_db_timestamp
time_t created
time_t last_modified
string created_str
string last_modified_str
vector<Handle_Type> handle_vector
vector<Irods_AVU_Type> avu_vector
vector<unsigned long int> handle_id_vector
vector<unsigned long int> handle_value_id_vector
vector<string> handle_name_string_vector
unsigned long int dublin_core_metadata_id
unsigned long int dublin_core_metadata_irods_object_id
unsigned long int irods_object_ref_id
```

```
bool encrypted
bool signed_gpg
bool compressed_tar_file
bool compressed_gzip
bool compressed_bzip2
unsigned long int detached_signature_irods_object_id
unsigned int gpg_key_pair_id_encrypt
unsigned int gpg_key_pair_id_sign
string gpg_key_fingerprint_encrypt
string gpg_key_fingerprint_sign
```

18.1.2 Member Functions

`void Irods_Object_Type (void)` [Default constructor]

`void Irods_Object_Type (unsigned int` *`uuser_id`*`, string` *`ppath`*`)` [Constructor]

`int set (unsigned int` *`uuser_id`*`, string` *`ppath`*`, unsigned long int` *`dc_metadata_id`* `= 0UL, unsigned long int` *`iirods_object_ref_id`* `= 0UL)` [Function]

`void clear (void)` [Function]

`void show (string` *`s`* `= "", stringstream *`*`strm`* `= 0)` [Function]

`int write_to_database (MYSQL *`*`mysql_ptr`*`)` [Function]

`int get_from_database (MYSQL *`*`mysql_ptr`*`, bool` *`id_only`* `= false)` [Function]

`int update (MYSQL *`*`mysql_ptr`*`)` [Function]

`int put_irods_object (string` *`filename`*`, string` *`irods_env_filename`*`, bool` *`force`* `= false)` [Function]

`int add_avu (Irods_AVU_Type` *`avu`*`, string` *`irods_env_filename`*`, bool` *`call_imeta`* `= true, bool` *`push_onto_vector`* `= true, bool` *`database`* `= true, MYSQL *`*`mysql_ptr`* `= 0, int` *`thread_ctr`* `= 0)` [Function]

```
int add_avu_cond (string irods_env_filename,                    [Function]
        string check_attrib,
        string check_val,
        string new_attrib,
        MYSQL *mysql_ptr = 0,
        string new_val = "",
        bool database = true,
        bool push_onto_vector = true,
        int thread_ctr = 0)
```

```
int delete_from_archive (MYSQL *&mysql_ptr,                     [Function]
        string thread_str = "")
```

```
int delete_from_gwirdsif_db (MYSQL *&mysql_ptr,                 [Function]
        string thread_str = "")
```

```
int add_handle_value (Handle_Type &handle,                      [Function]
        MYSQL *&mysql_ptr,
        string old_type_str,
        string path,
        unsigned int index,
        string new_type_str,
        string data_str,
        string thread_str = "")
```

```
const vector<Irods_AVU_Type>::iterator find_avu (string         [Function]
        attrib,
        string val,
        bool match_attrib_only = false,
        string thread_str = "")
```

```
int get_avus_from_irods_system (string command,                 [Function]
        string filename,
        Scan_Parse_Parameter_Type &param,
        int *ctr = 0)
```

```
int mark_for_deletion ( vector<Irods_Object_Type>               [Static function]
        &irods_object_vector,
        MYSQL *&mysql_ptr,
        Response_Type &response,
        int user_id,
        string irods_env_filename,
        time_t &save_delay,
        string thread_str = "",
        bool wake_purge_thread = false)
```

```
int undelete_irods_objects ( vector<Irods_Object_Type>      [Static function]
        &irods_object_vector,
        MYSQL *&mysql_ptr,
        bool archive,
        bool database,
        vector<Response_Type> &response_vector,
        string irods_env_filename,
        string thread_str = "")
```

```
int rename_irods_object (MYSQL *mysql_ptr,                  [Function]
        string irods_env_filename,
        Response_Type &response,
        vector<Response_Type> & new_response_vector,
        string new_filename,
        string thread_str = "")
```

```
int modify_irods_avu (string irods_env_filename,            [Function]
        string attribute,
        string old_value,
        string new_value,
        string thread_str = "")
```

```
int verify_signature (MYSQL *mysql_ptr,                     [Function]
        Response_Type &response,
        Irods_Object_Type &curr_irods_object,
        string thread_str = "")
```

18.2 class Irods_AVU_Type

The classes `Irods_Object_Type` and `Scan_Parse_Parameter_Type` and the function `zzparse` are `friends` of `class Irods_AVU_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, and Section 18.1 [class Irods_Object_Type], page 122.

18.2.1 Data Members

```
unsigned long int id                                        [Private variables]
int user_id
unsigned long int irods_object_id
bool deleted_from_archive
bool deleted_from_gwirdsif_db
string irods_object_path
string attribute
string value
string units
unsigned int time_set
```

18.2.2 Member Functions

```
void Irods_AVU_Type (void)                                  [Default constructor]
```

```
void Irods_AVU_Type (string attrib,                                    [Constructor]
        string val,
        string u = "",
        unsigned int t = 0U,
        unsigned int iirods_object_id = 0U,
        int uuser_id = 0,
        bool ddeleted_from_archive = false,
        bool ddeleted_from_gwirdsif_db = false)
void Irods_AVU_Type (const Irods_AVU_Type &i)                          [Constructor]
const Irods_AVU_Type& operator= (const                                 [Assignment operator]
        Irods_AVU_Type &i)
void set (string attrib,                                               [Function]
        string val,
        string u = "",
        unsigned int t = 0U,
        unsigned int iirods_object_id = 0U,
        unsigned int iid = 0U,
        string iirods_object_path = "",
        int uuser_id = 0,
        bool ddeleted_from_archive = false,
        bool ddeleted_from_gwirdsif_db = false)
int write_to_database (MYSQL *mysql_ptr, int thread_ctr = 0)           [Function]
void clear (void)                                                      [Function]
unsigned long int get_id (void)                                        [const inline functions]
bool get_deleted_from_archive (void)
bool get_deleted_from_gwirdsif_db (void)
int delete_irods_avu (Irods_AVU_Type &irods_avu,                       [Static function]
        bool delete_from_database = true,
        MYSQL *mysql_ptr = 0)
void show (string s = "", stringstream *strm = 0)                      [const function]
```

19 Handle Types

`class Handle_Type` and `class Handle_Value_Type` represent *handles* within the programs belonging to the `gwrdifpk` package. The data members correspond to the fields of the table `handles` in the `handlesystem` and `handlesystem_standalone` databases. See Chapter 27 [handlesystem and handlesystem_standalone Databases], page 158, and Chapter 3 [Standalone handle service], page 9.

`struct Handle_Value_Triple` is used as a convenience; it contains the data from three fields of a row from the `handles` database table, namely `idx`, `type` and `data`. The latter is stored as a C++ `string` in the data member `data_str`. See Section 19.3 [struct Handle_Value_Triple], page 133, below.

19.1 class Handle_Type

`class Scan_Parse_Parameter_Type` and the functions `main`, `exchange_data_with_client`, `generate_pids`, `Irods_Object_Type::mark_for_deletion`, `Irods_Object_Type::delete_from_archive`, `Irods_Object_Type::delete_from_gwirdsif_db`, `Irods_Object_Type::add_handle_value` and `purge_dc_metadata` are `friends` of `class Handle_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, and Section 18.1.2 [Irods_Object_Type Member Functions], page 123.

19.1.1 Data Members

`string handle` [Private variables]
`unsigned long int handle_id`
`map<unsigned long int, Handle_Value_Type> handle_value_map`

19.1.2 Member Functions

`void Handle_Type (void)` [Default constructor]
`void Handle_Type (const Handle_Type &h)` [Copy constructor]

`void operator= (const Handle_Type &h)` [Assignment operator]

`void clear (void)` [Function]

`int add_values (MYSQL *mysql_ptr,` [Functions]
`vector<Handle_Value_Triple> hvt_vector,`
`int user_id`
`vector<Handle_Value_Triple> *return_hvt_vector = 0,`
`bool lock_tables = true)`
`int add_value (MYSQL *mysql_ptr,`
`int iidx,`
`string ttype,`
`string ddata_str,`
`int user_id`
`Handle_Value_Triple *return_hvt = 0,`
`bool lock_tables = true)`

`int delete_from_database (MYSQL *mysql_ptr,` [Function]
`int user_id,`
`unsigned int options,`
`unsigned long int delay_value = 0UL,`
`int *handle_rows = 0,`
`int thread_ctr = 0)`

`const map<unsigned long int, Handle_Value_Type>::iterator` [Function]
`find (string type)`

`int created_by_user_id (void)` [Function]

`int fetch_handles_from_database (MYSQL *&mysql_ptr,` [Static functions]
`vector<.*> &handle_id_vector,`
`vector<.*> &handle_vector,`
`string type = "",`
`string thread_str = "")`
`int fetch_handle_from_database (MYSQL *&mysql_ptr,`
`unsigned long int handle_id,`
`Handle_Type &handle,`
`string type = "",`
`string thread_str = "")`

`int delete_handle_values (std::set<unsigned long int>` [Static function]
`&handle_id_set,`
`vector<.*> &handle_value_type_vector,`
`MYSQL *&mysql_ptr,`
`string thread_str = "",`
`bool lock_table = true)`

`int delete_handle_values (vector<.*>` [Function]
`&handle_value_type_vector,`
`MYSQL *&mysql_ptr,`
`string thread_str = "",`
`bool lock_table = true)`

`int unmark_handle_for_deletion (MYSQL *&mysql_ptr,` [Function]
`string thread_str = "")`

`void show` [const function]
`(string s = "Handle_Type:", stringstream * strm = 0)`

19.2 class Handle_Value_Type

`class Handle_Value_Type` is declared in '`hndlvltp.web`'.

The classes `Scan_Parse_Parameter_Type`, `Irods_Object_Type`, and `Handle_Type` and the functions `main`, `exchange_data_with_client`, `zzparse`, `generate_pids` and `purge_dc_metadata` are `friends` of `class Handle_Value_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, Section 18.1 [class Irods_Object_Type], page 122, Section 19.1 [class Handle_Type], page 127, Section 26.5 [Exchanging data], page 151, Section 26.6 [Scanning and parsing], page 152, and Section 26.8 [Generating PIDs], page 153.

19.2.1 Data Members

`string filename` [Private variables]
`string handle`
`int idx`
`string type`
`char* data`
`unsigned int data_length`
`int ttl_type`
`int ttl`
`time_t timestamp`
`char* refs`
`unsigned int refs_length`
`bool admin_read`
`bool admin_write`
`bool pub_read`
`bool pub_write`
`unsigned long int handle_id`
`unsigned long int handle_value_id`
`int created_by_user_id`
`string created_by_user_name`
`time_t created`
`time_t last_modified`
`string created_str`
`string last_modified_str`
`bool marked_for_deletion`
`time_t delete_from_database_timestamp`
`string delete_from_database_timestamp_str`
`unsigned long int irods_object_id`

Public static `unsigned int` constants in `Handle_Value_Type`:

Name	Value
`NULL_HANDLE_VALUE_TYPE_INDEX`	0
`IRODS_OBJECT_INDEX`	1
`IRODS_OBJECT_PID_INDEX`	11
`IRODS_OBJECT_REF_INDEX`	21
`IRODS_OBJECT_REF_PID_INDEX`	22
`IRODS_OBJECT_DELETED_FROM_ARCHIVE_INDEX`	31
`IRODS_OBJECT_DELETED_FROM_GWIRDSIF_DB_INDEX`	41

CHECKSUM_SHA224_INDEX	303
CHECKSUM_SHA256_INDEX	304
CHECKSUM_SHA384_INDEX	305
CHECKSUM_SHA512_INDEX	306
ENCRYPTED_INDEX	401
SIGNED_INDEX	402
CLEARSIGNED_INDEX	403
VERIFIED_INDEX	404
GPG_KEY_FINGERPRINT_INDEX	405
DETACHED_SIGNATURE_INDEX	406
DETACHED_SIGNATURE_REF_INDEX	407
DETACHED_SIGNATURE_PID_INDEX	408
COMPRESSED_TAR_FILE_INDEX	501
COMPRESSED_GZIP_INDEX	502
COMPRESSED_BZIP2_INDEX	503
RESERVED_0_INDEX	601
RESERVED_1_INDEX	701
RESERVED_2_INDEX	801
OTHER_HANDLE_VALUE_TYPE_INDEX	901

`map<int, string> idx_type_map` [Public static variables]
`map<string, unsigned int> type_idx_map`

19.2.2 Member Functions

`void Handle_Value_Type (void)` [Default constructor]
`void Handle_Value_Type (const Handle_Value_Type &h)` [Copy constructor]

`void ~Handle_Value_Type (void)` [Destructor]

```
void operator= (const Handle_Value_Type &h)
```
[Assignment operator]

```
int initialize_maps (void)
```
[Static function]

```
int set (MYSQL_ROW &curr_row,
        unsigned int field_ctr,
        string hhandle = "",
        int thread_ctr = -1)
int set (string hhandle,
        int iidx,
        string ttype,
        char *ddata,
        unsigned int ddata_length,
        int tttl_type,
        int tttl,
        time_t ttimestamp,
        char *rrefs,
        unsigned int rrefs_length,
        bool aadmin_read,
        bool aadmin_write,
        bool ppub_read,
        bool ppub_write,
        long int ccreated_by_user_id,
        unsigned long int hhandle_id,
        unsigned long int hhandle_value_id,
        bool mmarked_for_deletion,
        time_t ccreated,
        time_t llast_modified,
        string ccreated_str = "",
        string llast_modified_str = "",
        string ffilename = "")
```
[Functions]

```
int delete_handle_value (MYSQL *&mysql_ptr,
        string hv_str,
        deque<Response_Type> &response_deque,
        int user_id,
        string username,
        unsigned int privileges,
        unsigned int options = 0U,
        unsigned long int delay_value = 0,
        string thread_str = "")
```
[Static Function]

```
int unmark_handle_value_for_deletion (                    [Static Function]
        MYSQL *&mysql_ptr,
        string hv_str,
        deque<Response_Type> &response_deque,
        int user_id,
        string username,
        unsigned int privileges,
        string thread_str = "")
int extract (string hv_str,                               [Static Function]
        string &prefix,
        string &sub_handle,
        string &handle,
        int &index,
        string &type,
        string thread_str = "")
int write_to_database (MYSQL *mysql_ptr,                  [Function]
        string database,
        bool replace = false)
void clear (void)                                         [Function]
void show (string s = "Handle_Value_Type:",               [const Function]
        stringstream *strm = 0)
```

19.3 struct Handle_Value_Triple

19.3.1 Data Members

```
int idx                                                   [Public variables]
string type
string data_str
```

19.3.2 Member Functions

```
void Handle_Value_Triple (void)                           [Default constructor]
void Handle_Value_Triple (int iidx,                       [Constructor]
        string ttype,
        string ddata_str = "")
const Handle_Value_Triple& operator= (                    [Assignment operator]
        const Handle_Value_Triple &hvt)
void clear (void)                                         [Function]
int show (string s = "")                                  [Function]
```

20 X.509 Certificate Types

See Section 28.3 [Certificates database table], page 161.

20.1 class X509_Cert_Type

`class X509_Cert_Type` represents *X.509 certificates* within `gwrdifpk`. It contains data members corresponding to the fields of an X.509 certificate and to the columns of the database table `gwirdsif.Certificates`. See Section 28.3 [Certificates], page 161.

The following classes and functions are `friends` of `class X509_Cert_Type`:

`class Scan_Parse_Parameter_Type`
: See Chapter 14 [class Scan_Parse_Parameter_Type], page 107.

`class Distinguished_Name_Type`
: See Section 20.2 [class Distinguished_Name_Type], page 136, below.

`class User_Info_Type`
: See Chapter 16 [class User_Info_Type], page 119.

`get_user_info_func`
: See Section 26.7 [Parser rule functions], page 153.

`extract_dn_fields`
: See Section 26.4 [X.509 certificates], page 151.

`verify_certificate`
: See Section 26.4 [X.509 certificates], page 151.

`yyparse` See Section 26.6 [Scanning and parsing], page 152.

20.1.1 Data members

[Private variables]

```
string organization
string organizationalUnitName
string commonName
string countryName
string localityName
string stateOrProvinceName
unsigned long int serialNumber
time_t Validity_notBefore
time_t Validity_notAfter
X509_Cert_Type* issuer_cert
string user_name
unsigned int user_id
unsigned int certificate_id
unsigned int issuer_cert_id
bool is_ca
bool is_proxy
```

20.1.2 Member functions

`void X509_Cert_Type (void)` [Default constructor]

`void X509_Cert_Type (const X509_Cert_Type &cert)` [Copy constructor]

```
void X509_Cert_Type (unsigned long int sserialNumber,          [Constructor]
        X509_Cert_Type *iissuer_cert = 0,
        string oorganization = "",
        string oorganizationalUnitName = "",
        string ccommonName = "",
        string ccountryName = "",
        string llocalityName = "",
        string sstateOrProvinceName = "",
        unsigned int uuser_id = 0,
        string uuser_name = "",
        time_t VValidity_notBefore = 0,
        time_t VValidity_notAfter = 0,
        bool iis_ca = false,
        bool iis_proxy = false,
        unsigned int ccertificate_id = 0,
        unsigned int iissuer_cert_id = 0)
```

```
int set (unsigned long int sserialNumber,                      [Function]
        X509_Cert_Type *iissuer_cert = 0,
        string oorganization = "",
        string oorganizationalUnitName = "",
        string ccommonName = "",
        string ccountryName = "",
        string llocalityName = "",
        string sstateOrProvinceName = "",
        unsigned int uuser_id = 0,
        string uuser_name = "",
        time_t VValidity_notBefore = 0,
        time_t VValidity_notAfter = 0,
        bool iis_ca = false,
        bool iis_proxy = false,
        unsigned int ccertificate_id = 0,
        unsigned int iissuer_cert_id = 0)
```

`int set (MYSQL_ROW &row, string thread_ctr_str = "")` [Function]

`void operator= (const X509_Cert_Type &cert)` [Assignment operator]

```
int get_from_database (MYSQL *mysql_ptr, unsigned int          [Function]
        uuser_id, string thread_ctr_str = "")
```

`void clear (void)` [Function]

```
void show (string s = "",                                    [const function]
        stringstream *strm = 0,
        bool show_issuer = false)
string output (void)                                         [const function]
```

20.2 class Distinguished_Name_Type

`class Distinguished_Name_Type` represents the *distinguished name* from an *X.509 certificate* within `gwrdifpk`. See Section 5.3 [X.509 Certificates], page 62, Section 20.1 [class X509_Cert_Type], page 134, above, and Section 28.3 [Certificates (database table)], page 161.

`class Scan_Parse_Parameter_Type` is a `friend` of `Distinguished_Name_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107.

20.2.1 Data members

```
string organization                                          [Private variables]
string organization
string organizationalUnitName
string commonName
string countryName
string stateOrProvinceName
string localityName
string user_name
int user_id
```

20.2.2 Member functions

```
void Distinguished_Name_Type (void)                          [Default constructor]
void Distinguished_Name_Type (string oorganization,          [Constructor]
        string oorganizationalUnitName = "",
        string ccommonName = "", ,
        string ccountryName = "", ,
        string llocalityName = "",
        string sstateOrProvinceName = "",
        unsigned int uuser_id = 0,
        string uuser_name = "")
int set (string distinguished_name_str,                      [Function]
        int uuser_id = -1,
        string *uuser_name = 0)
void operator= (const X509_Cert_Type& c)                     [Assignment operator]
bool operator== (const                                       [Equality operators (constant)]
        Distinguished_Name_Type &d)
bool operator== (const string &s)
bool operator!= (const                                       [Inequality operator (constant)]
        Distinguished_Name_Type&d)
```

`void clear (void)` [Function]

`string output (void)` [Function]

`void show (string` *`s`* `= "", stringstream *`*`strm` *`*`* `= 0)` [const function]

21 Dublin Core Metadata Types

`class Dublin_Core_Metadata_Type` and `class Dublin_Core_Metadata_Sub_Type` represent *metadata* in XML format corresponding to the *Dublin Core* standard. The data members of these classes correspond to the fields in the tables `Dublin_Core_Metadata` and `Dublin_Core_Metadata_Sub` in the `gwirdsif` database table. See Section 28.13 [Dublin Core database tables (gwirdsif)], page 170.

The client program `gwirdcli` sends Dublin Core metadata to the server program `gwirdsif` in the form of a file containing XML code. See Section 8.6 [User commands for Dublin Core metadata], page 84. `gwirdsif` uses the `expat` library to parse the XML data. See Section 2.2 [Prerequisites], page 4.

21.1 class Dublin_Core_Metadata_Type

`class Scan_Parse_Parameter_Type` and the functions `exchange_data_with_server`, `exchange_data_with_client`, `yyparse` and `purge_dc_metadata` are `friends` of `Dublin_Core_Metadata_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, Section 26.5 [Exchanging data], page 151, and Section 26.6 [Scanning and parsing], page 152.

21.1.1 Data Members

[Private variables]

```
unsigned long int id
unsigned int user_id
unsigned int irods_server_id
string irods_object_path
string dc_metadata_irods_object_path
unsigned long int handle_id
unsigned long int irods_object_ref_id
unsigned long int irods_object_self_id
string created_str
time_t created
string last_modified_str
time_t last_modified
bool marked_for_deletion
int delete_file
string delete_from_database_timestamp_str
time_t delete_from_database_timestamp
multimap<unsigned int, Dublin_Core_Metadata_Sub_Type>
        metadata_sub_map
stack<Dublin_Core_Metadata_Sub_Type> metadata_sub_stack
```

[Public static variables]

```
map<unsigned int, string> element_map
map<string, unsigned int> element_ctr_map
map<unsigned int, string> qualifier_map
map<string, unsigned int> qualifier_ctr_map
```

Public static unsigned int constants in `class Dublin_Core_Metadata_Type`:

Name	Value
DUBLIN_CORE_NULL_TYPE	0
DUBLIN_CORE_ELEMENT_TYPE	1
DUBLIN_CORE_QUALIFIER_TYPE	2
DUBLIN_CORE_ATTRIBUTE_TYPE	3

Elements:

Name	Value
DUBLIN_CORE_NULL_ELEMENT	0
DUBLIN_CORE_TITLE_ELEMENT	1
DUBLIN_CORE_CREATOR_ELEMENT	2
DUBLIN_CORE_SUBJECT_ELEMENT	3
DUBLIN_CORE_DESCRIPTION_ELEMENT	4
DUBLIN_CORE_PUBLISHER_ELEMENT	5
DUBLIN_CORE_CONTRIBUTOR_ELEMENT	6
DUBLIN_CORE_DATE_ELEMENT	7
DUBLIN_CORE_TYPE_ELEMENT	8
DUBLIN_CORE_FORMAT_ELEMENT	9
DUBLIN_CORE_IDENTIFIER_ELEMENT	10
DUBLIN_CORE_SOURCE_ELEMENT	11
DUBLIN_CORE_LANGUAGE_ELEMENT	12
DUBLIN_CORE_RELATION_ELEMENT	13
DUBLIN_CORE_COVERAGE_ELEMENT	14
DUBLIN_CORE_RIGHTS_ELEMENT	15

Terms:

Name	Value
DUBLIN_CORE_NULL_TERM	100
DUBLIN_CORE_ABSTRACT_TERM	101
DUBLIN_CORE_ACCESSRIGHTS_TERM	102
DUBLIN_CORE_ACCRUALMETHOD_TERM	103
DUBLIN_CORE_ACCRUALPERIODICITY_TERM	104
DUBLIN_CORE_ACCRUALPOLICY_TERM	105
DUBLIN_CORE_ALTERNATIVE_TERM	106
DUBLIN_CORE_AUDIENCE_TERM	107
DUBLIN_CORE_AVAILABLE_TERM	108
DUBLIN_CORE_BIBLIOGRAPHICCITATION_TERM	109
DUBLIN_CORE_CONFORMSTO_TERM	110
DUBLIN_CORE_CONTRIBUTOR_TERM	111
DUBLIN_CORE_COVERAGE_TERM	112
DUBLIN_CORE_CREATED_TERM	113
DUBLIN_CORE_CREATOR_TERM	114
DUBLIN_CORE_DATE_TERM	115

21.1.2 Member Functions

`void Dublin_Core_Metadata_Type (void)` [Default constructor]

`void ~Dublin_Core_Metadata_Type (void)` [Destructor]

```
bool operator== (                                   [Equality operator (const)]
        const Dublin_Core_Metadata_Type &d)

int initialize_maps (void)                          [Static function]

int set (unsigned long int iid,                     [Function]
        unsigned long int uuser_id,
        unsigned int iirods_server_id,
        string iirods_object_path,
        string ddc_metadata_irods_object_path,
        unsigned long int hhandle_id,
        unsigned long int iirods_object_ref_id,
        unsigned long int iirods_object_self_id,
        string ccreated_str,
        time_t ccreated,
        string llast_modified_str,
        time_t llast_modified,
        bool mmarked_for_deletion,
        int ddelete_file,
        string ddelete_from_database_timestamp_str,
        time_t ddelete_from_database_timestamp)

void XMLCALL start (void *data,                     [Static function]
        const char *el,
        const char **attr)

void XMLCALL end (void *data, const char *el)       [Static function]

void handle_data (void *data,                       [Static function]
        const char *content,
        int length)

int output (ofstream &out_strm)                     [const function]

int parse (char *buffer)                            [Function]

void clear (void)                                   [Function]

int show (string s = "",                            [const function]
        bool show_sub_map = true,
        ostream *out_strm = &cerr)

int write_dc_metadata_to_database (MYSQL *mysql_ptr,    [Function]
        unsigned long int irods_object_ref_id = 0UL,
        bool replace = false,
        string database = "gwirdsif"

int set_handle_id (MYSQL *mysql_ptr,                [Function]
        unsigned long int hhandle_id)
```

```
int mark_dc_metadata_for_deletion (MYSQL *&mysql_ptr,          [Static function]
        Response_Type &response,
        deque<Response_Type> &response_deque,
        string irods_current_dir,
        int user_id,
        unsigned int &errors_occurred,
        unsigned int &warnings_occurred,
        string thread_str)
```

```
int undelete_dc_metadata (MYSQL *&mysql_ptr,                   [Static function]
        Response_Type &response,
        deque<Response_Type> &response_deque,
        string irods_current_dir,
        int user_id,
        unsigned int &errors_occurred,
        unsigned int &warnings_occurred,
        string thread_str)
```

```
int get_dc_metadata_from_database (MYSQL *&mysql_ptr,          [Static function]
        Response_Type &response,
        map<unsigned long int, Dublin_Core_Metadata_Type>
        &dc_metadata_map,
        vector<string> &irods_object_path_vector,
        vector<unsigned int> &id_vector,
        bool get_expired,
        unsigned int limit,
        string database = "gwirdsif",
        string thread_str = "")
```

21.2 class Dublin_Core_Metadata_Sub_Type

The classes `Scan_Parse_Parameter_Type` and `Dublin_Core_Metadata_Type` and the the functions `yyparse`, `exchange_data_with_server` and `exchange_data_with_client` are `friends` of `class` `Dublin_Core_Metadata_Sub_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, Section 21.1 [class Dublin_Core_Metadata_Type], page 138, Section 26.6 [Scanning and parsing], page 152, and Section 26.5 [Exchanging data], page 151.

21.3 Dublin_Core_Metadata_Sub_Type Data Members

```
unsigned long int id                                   [Private data members]
unsigned long int metadata_id
unsigned int element_id
unsigned int term_id
unsigned int qualifier_id
string value
multimap <string, string> attribute_map
```

21.4 Dublin_Core_Metadata_Sub_Type Functions

`void Dublin_Core_Metadata_Sub_Type (void)` [Default constructor]

`void ~Dublin_Core_Metadata_Sub_Type (void)` [Destructor]

`bool operator== (const Dublin_Core_Metadata_Sub_Type &d)` [Equality operator (const)]

`int set (unsigned long int iid, unsigned long int mmetadata_id, unsigned int eelement_id, unsigned int qqualifier_id, unsigned int tterm_id, string vvalue, multimap <string, string> *attribute_map_ptr = 0, string thread_str = "")` [Function]

`int set (MYSQL *&mysql_ptr, MYSQL_ROW &curr_row, string database = "gwirdsif", string thread_str = "")` [Function]

`bool operator!= (const Dublin_Core_Metadata_Sub_Type &d)` [Inequality operator (const)]

`void clear (void)` [Function]

`int show (string s = "")` [const function]

22 GPG_Key_Pair_Type

`class GPG_Key_Pair_Type` is defined in '`gpgkprtp.web`'. `class Scan_Parse_Parameter_Type` and the functions `yyparse` and `verify_gpg_signature` are `friends` of `GPG_Key_Pair_Type`.

22.1 Data Members

`unsigned int gpg_key_pair_id` [Private variables]
`int user_id`
`string uid`
`string fingerprint`
`string public_key`
`bool revoked`
`time_t created`
`string created_str`
`time_t last_modified`
`string last_modified_str`

22.2 Member Functions

`GPG_Key_Pair_Type (void)` [Default constructor]

`GPG_Key_Pair_Type (const GPG_Key_Pair_Type &g)` [Copy constructor]

`~GPG_Key_Pair_Type (void)` [Destructor]

`string get_key_id (void)` [Function]

`int get_gpg_key_pair_from_database (MYSQL *mysql_ptr,` [Function]
`string key_id = "",`
`bool get_public_key = false,`
`string thread_str = "")`

`void clear (void)` [Function]

`int show (string s = "")` [const function]

23 class Pull_Request_Type

`class Scan_Parse_Parameter_Type` and the function `yyparse` are `friend`s of `class Pull_Request_Type`.

See also Section 28.15 [Pull Request database table (gwirdsif)], page 172.

23.1 Data Members

Private data members:

`int pull_request_id` [Variable]

`int user_id` [Variable]

`string username` [Variable]

`string distinguished_name` [Variable]

`string server_hostname` [Variable]

`string server_ip_address` [Variable]

`string client_hostname` [Variable]

`string client_ip_address` [Variable]

`unsigned int client_port` [Variable]

`string client_port_str` [Variable]

`int pull_interval` [Variable]

`time_t latest_pull` [Variable]

`time_t created` [Variable]

`time_t last_modified` [Variable]

`bool force_flag` [Variable]

Public data members:

`int DEFAULT_PULL_INTERVAL` [Variable]
Value: 86400 = 1 day in seconds

The following static constants are used for setting and testing bit-fields. In particular, they are used when invoking the function `Pull_Request_Type::update_pull_request`. See Section 23.2 [Pull_Request_Type Member Functions], page 146.

`unsigned int LATEST_PULL` [Static constant]
Value: 1

`unsigned int CREATED` [Static constant]
Value: 2

`unsigned int LAST_MODIFIED` [Static constant]
Value: 4

23.2 Member Functions

`void Pull_Request_Type (void)` [Default constructor]

`const Pull_Request_Type& operator= (const Pull_Request_Type &p)` [Assignment operator]

`void clear (void)` [Function]

`void show (string s = "")` [const function]

`int write_pull_request_to_database (MYSQL *mysql_ptr, string thread_str = "")` [Function]

`int get_pull_request_from_database (MYSQL *mysql_ptr, int &return_pull_request_id, bool id_only = false, bool lock_tables = true, bool assign = false, string thread_str = "")` [Function]

`int get_pull_requests_from_database (MYSQL *mysql_ptr, vector<int> pull_request_id_vector, vector<int> user_id_vector, vector<string> username_vector, vector<string> client_hostname_vector, vector<string> client_ip_address_vector, vector<Pull_Request_Type> &pull_request_vector, bool expired = false, bool id_only = false, bool lock_tables = true, string thread_str = "")` [static function]

`int get_expired (MYSQL *mysql_ptr, vector<Pull_Request_Type> &pull_request_vector, bool lock_tables = true, string thread_str = "")` [static function]

`int contact_pull_client (MYSQL *&mysql_ptr, const gnutls_certificate_credentials_t &xcred, string thread_str = "")` [Function]

`int update_pull_request (MYSQL *&mysql_ptr, unsigned int fields, string thread_str = "")` [Function]

24 class Pull_Response_Type

`class Pull_Response_Type`. `class Scan_Parse_Parameter_Type` and the functions `pull_response` and `xxparse` are `friends` of `class Pull_Response_Type`. See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, Section 26.14 [Pull functions], page 155, and Section 26.6 [Scanning and parsing], page 152. See also Section 29.7 [Pull Response database tables (gwirdcli)], page 175.

24.1 Data Members

24.1.1 Private data members

`int pull_response_id` [Variable]

`int user_id` [Variable]

`string username` [Variable]

`string server_hostname` [Variable]

`string server_ip_address` [Variable]

`string client_hostname` [Variable]

`string client_ip_address` [Variable]

`string Distinguished_Name` [Variable]

`int pull_interval` [Variable]

`time_t latest_pull` [Variable]

`time_t created` [Variable]

`time_t last_modified` [Variable]

`bool force_flag` [Variable]

`int pull_request_id` [Variable]

`vector<Pull_Path_Type> pull_path_vector` [Variable]

24.1.2 Public static constant data members

`int DEFAULT_PULL_INTERVAL` [Static constant]

Value: 86400, i.e., one day in seconds.

`unsigned int LATEST_PULL` [Static constants]
`unsigned int PULL_INTERVAL`
`unsigned int CREATED`
`unsigned int LAST_MODIFIED`
`unsigned int CHECKSUMS`

These static constants have the values 1, 2, 4, 8 and 16, i.e., $2^0 \ldots 2^4$ and are used with bit-fields.

24.2 Member Functions

`void Pull_Response_Type (`*void*`)` [Default constructor]

`int set (string` *s*`,` [Function]
`const string &`*server_ip_address*`,`
`string` *thread_str* `= "")`

`const Pull_Response_Type& operator= (const` [Assignment operator]
`Pull_Response_Type &`*p*`)`

`void clear (void)` [Function]

`void show (string` *s* `= "")` [Function]

`int get_pull_response_from_database (MYSQL *&`*mysql_ptr*`,` [Function]
`bool` *expired* `= false,`
`bool` *create_new* `= false,`
`string` *thread_str* `= "")`

`int update_database (unsigned int` *flags*`,` [Function]
`Scan_Parse_Parameter_Type &`*param*`,`
`int` *int_val* `= 0,`
`string` *thread_str* `= "")`

`int write_pull_response_to_database (MYSQL *&`*mysql_ptr*`,` [Function]
`bool` *lock* `= false,`
`string` *thread_str* `= "")`

25 Miscellaneous Types

25.1 struct Initialize_Exception_Type

`struct Initialize_Exception_Type` has an "empty" declaration. That is, it has neither data members nor explicitly defined member functions. It is *thrown* by constructors if an error occurs that prevents the object from being constructed properly. Normally in the `gwrdifpk` package, functions communicate that an error has occurred to their callers by means of their return values. However, in C++, constructors don't have return values, so that exceptions or some other means must be used.

25.2 class Cookie_Type

The function `main` is a `friend` of `class Cookie_Type`. `class Cookie_Type` is only used in the web application '`gwrdwbap`'.

25.3 Global variables

`pthread_mutex_t cookie_vector_mutex` [Global variables]
`vector <Cookie_Type> cookie_vector`

25.3.1 Cookie_Type Data Members

`string name` [Private variables]
`string value`
`string domain`
`string path`
`string expires_str`
`time_t expires`
`Scan_Parse_Parameter_Type* param`
`vector<pair<string, string> > name_value_pairs`

25.3.2 Cookie_Type Member Functions

`void Cookie_Type (void)` [Default constructor]

`void Cookie_Type (const Cookie_Type &c)` [Copy constructor]

`void operator= (const Cookie_Type &c)` [Assignment operator]

`int operator== (const Cookie_Type &c)` [Equality operator (const)]

`bool operator< (const Cookie_Type &c)` [Less-than operator (const)]

`void clear (void)` [Function]

`void show (string s = "Cookie_Type:", stringstream *t = 0)` [const function]

`int parse_cookies (const char *http_cookie_str, string &session_id, stringstream &out_strm)` [static function]

`int generate_session_id (string& session_id,` [static function]
`stringstream &out_strm)`

`void* delete_expired_cookies (void *v)` [static thread function]

Thread function. Called in `main` of the web application '`gwrdwbap`'.

26 Non-class Functions

26.1 Main (and similar) functions

`int main (int argc, char *argv[])` [Main functions]

`int client_func (Scan_Parse_Parameter_Type &param)` [Client function]

26.2 Process command-line options

`int process_command_line_options (int argc, char *argv[])` [Function]

`gwirdsif` and `gwirdcli` both call this function to process their respective command-line arguments. This implies that they share the same options. This is done in order to help ensure consistency between the two programs. Some options have no meaning in one or the other of the programs, while others may be interpreted differently.

26.3 Listen and connect functions

`void* connect_func (void *v)` [Connection function]

`void* listen_remote_anon (void *v)` [Listen functions]
`void* void* listen_remote_X_509(void *v)`

`gnutls_session_t initialize_tls_session (void)` [Functions]
`int generate_dh_params (void)`

26.4 X.509 certificates

```
int extract_dn_fields (gnutls_x509_crt_t &cert,                    [Function]
        X509_Cert_Type *x509_cert = 0,
        bool subject = true,
        Scan_Parse_Parameter_Type *param = 0)
```

Defined in 'gntlsfnc.web'.

```
int verify_certificate (gnutls_session_t session,                  [Function]
        X509_Cert_Type *x509_cert_ptr = 0,
        const char *hostname = 0)
```

Defined in 'ex_rfc2818.web'.

26.5 Exchanging data

```
int exchange_data_with_client (Scan_Parse_Parameter_Type           [Function]
        &param)
```

Defined in '[...]/Finston/gwrdifpk/src/exchncli.web'.

```
int exchange_data_with_server (Scan_Parse_Parameter_Type           [Function]
        &param)
```

Defined in '[...]/Finston/gwrdifpk/src/exchnsrv.web'.

These are two of the most important functions in `gwrdifpk`. They are responsible for the communication between the two *peers*, i.e., the server program `gwirdsif` and the client program `gwirdcli`. Each contains an *endless loop*. At the beginning, `gnutls_record_recv` or `recv` (depending on the kind of connection) is called in order to receive any data sent from the peer. If there is any, it's passed to the *parser function*, `yyparse` on the server-side or `zzparse` on the client-side. See Section 26.6 [Scanning and parsing], page 152. After parsing, the variable `Scan_Parse_Parameter_Type::response_deque` belonging to the `Scan_Parse_Parameter_Type` object used for the current connection is examined. If it contains any *responses*, i.e., objects of type `Response_Type`, the latter are processed. These may cause result in data being sent to the peer. Then the process is repeated, i.e., the loop repeats. (See Chapter 14 [class Scan_Parse_Parameter_Type], page 107, and Chapter 15 [class Response_Type], page 116.)

In both the client and the server, the variables `bool client_finished` and `bool server_finished` are defined. Communication ends when both of these variables have the value `true`. Either peer may break off the connection.

26.6 Scanning and parsing

`int yyparse (yyscan_t parameter)` [Parser functions]
`int zzparse (yyscan_t parameter)`
`int xxparse (yyscan_t parameter)`

These functions are generated by GNU Bison from the input files '`gwrdifpk-1.0/src/parser.web`', '`gwrdifpk-1.0/src/prsrclnt.web`' and '`gwrdifpk-1.0/src/prsrcln2.web`', respectively. See Section 2.2 [Prerequisites], page 4.

`yyparse` implements the grammar of the commands "understood" by the server while `zzparse` serves the same purpose for the client. Please note that the user commands are sent by the client to the server and interpreted by the latter. `zzparse` only ever interprets commands sent back to the client from the server, so users never pass these commands to the client.

`xxparse` implements the grammar of the commands used to control the client.

`int yylex (YYSTYPE *lvalp, yyscan_t parameter)` [Scanner functions]
`int zzlex (YYSTYPE *lvalp, yyscan_t parameter)`
`int xxlex (YYSTYPE *lvalp, yyscan_t parameter)`

These functions are generated by Flex from the input files `gwrdifpk-1.0/src/scanner.web`, `gwrdifpk-1.0/src/scnrclnt.web` and `gwrdifpk-1.0/src/scnrcln2.web`, respectively. See Section 2.2 [Prerequisites], page 4.

`yylex` implements the server-side *lexical scanner*, while `zzlex` and `xxlex` implement the two client-side scanners. They are called by the *parser functions* `yyparse`, `zzparse` and `xxparse`, respectively. See above.

The server and client each receive input in the form of a stream of bytes (characters). The *parser function*, i.e., `yyparse` for the server and `zzparse` for the client, calls the scanner function, i.e., `yylex` or `zzlex`, repeatedly. The latter converts the stream of bytes successively into *tokens* according to rules defined in the Flex input files. The tokens, along with their *semantic values*, if any, are passed back to the scanner function.

`int yywrap (void)` [Additional functions for parsers]
`int zzwrap (void)`
`int xxwrap (void)`
`int yyerror (void *v, char const *s)`
`int zzerror (void *v, char const *s)`
`int xxerror (void *v, char const *s)`

26.7 Parser rule functions

These functions are defined in '`prsrfncs.web`' and are called in `yyparse`. See Section 26.6 [Scanning and parsing], page 152, above. That is, they are called in the *parser rules* defined in '`parser.web`'.

It can be advantageous to put the code for complicated actions into a separate function, rather than to include it in a parser rule, in order to decrease compilation time.

Of course, it would be possible to define functions that are called in rules for the client-side parser `yyparse`. However, at the present time, no such functions are defined in `gwrdifpk`.

`int client_sending_file_rule_func` [Function]
`(Scan_Parse_Parameter_Type* param,`
`string filename, int reference)`

`int distinguished_name_rule_func` [Function]
`(Scan_Parse_Parameter_Type* param, const char*`
`distinguished_name_str)`

`int get_user_info_func (Scan_Parse_Parameter_Type* param,` [Function]
`const char* curr_username = 0)`

26.8 Generating PIDs

`int generate_pids (MYSQL *ysql_ptrm,` [Function]
`string prefix_str,`
`string &pid_str,`
`vector<string> *pid_vector_ptr = 0,`
`unsigned int number_of_pids = 1,`
`vector<unsigned long int> *handle_id_vector_ptr = 0,`
`vector<unsigned long int> *handle_value_id_vector_ptr = 0,`
`bool standalone_hs = true,`
`string institute_str = "",`
`string suffix_str = "",`
`vector<Handle_Type> *handle_vector = 0,`
`string fifo_pathname = "",`
`long int user_id = 0,`
`string username = "")`

Defined in '`pidfncs.web`'.

`int check_prefix (MYSQL *mysql_ptr,` [Function]
`string prefix_str,`
`unsigned int &prefix_id)`
Defined in '`pidfncs.web`'.

26.9 Cryptographic operations

`int decrypt (string encrypted_text,` [Function]
`bool is_file,`
`char *plain_text,`
`size_t plain_text_length,`
`string output_filename = "",`
`char *passphrase = 0,`
`char end_char = '\n',`
`string thread_str = "")`

`int verify_gpg_signature (MYSQL *mysql_ptr,` [Function]
`string base_str,`
`string base_filename = "",`
`string detached_signature_str = "",`
`string detached_signature_filename = "",`
`string gpg_key_fingerprint = "",`
`GPG_Key_Pair_Type *return_gpg_key_pair = 0,`
`bool store_signature = false,`
`string *store_signature_filename = 0,`
`bool overwrite = false,`
`string thread_str = "")`

26.10 Deleting and rotating files

`void* purge_server_logs (void *v)` [Thread function]
Deletes old temporary files, rotates log files, and deletes expired database entries.

Old temporary files that are no longer needed may "pile up" in the '`/tmp/`' directory if the server program `gwirdsif` is invoked with the '`--save-temp-files`' option (see Chapter 6 [Invoking gwirdsif/gwirdcli], page 63) or if the program is killed by a signal.

Defined in '`gwrdifpk-1.0/src/purgfncs.web`'.

`int rotate_log_file (string &filename,` [Function]
`time_t now,`
`time_t limit,`
`pthread_mutex_t &mutex = 0,`
`ostream *out_strm = 0)`
Called by `purge_server`.

Defined in '`gwrdifpk-1.0/src/purgfncs.web`'.

`void* purge_irods_archive` [Thread function]
Defined in '`gwrdifpk-1.0/src/purgarch.web`'.

`void* purge_server_database (void *v)` [Thread function]
Defined in 'gwrdifpk-1.0/src/purgdtbs.web'.

`void* purge_dc_metadata (void *v)` [Thread function]
Defined in 'gwrdifpk-1.0/src/purgdcmd.web'. A conservative approach is taken to modifying the handles: Neither the handles nor the handle values of type `DC_METADATA` or `DC_METADATA_PID` are deleted! Instead, handle values of type `DC_METADATA_DELETED` and/or `DC_METADATA_DELETED_PID` are added to the existing handles.

Since handles are supposed to be "persistent identifiers", they should normally not be deleted. However, there is no reason not to delete *handle values*, if the information they contain becomes out-of-date. However, at this time, I think it's better to leave the old handle values and add new ones rather than replacing the old ones. They may be useful for searching. This may change at some future date.

26.11 Signal handlers

`void signal_handler (int sig)` [Signal handler]
Defined in 'sgnlhndl.web'. The main function of the server program `gwirdsif` sets this function to handle the signals '`SIGINT`' ("Interrupt") and '`SIGTERM`' ("Terminate"). See also Section 11.3 [Signal handling], page 99.

`void initialize_signal_maps (void)` [Function]

26.12 Exit handlers

`void finish_gwirdsif (void)` [Exit handler]
Defined in 'gwirdsif.web'.

`void finish_gwirdcli (void)` [Exit handler]
Defined in 'gwirdcli.web'.

26.13 Time functions

`string convert_seconds (time_t seconds)` [Function]

`unsigned long int convert_time_spec (string time_spec)` [Function]

```
int get_seconds_since_epoch (const char *timestamp,          [Function]
        time_t &sse,
        string format_str = "%Y-%m-%d %H:%M:%S")
```

```
string get_datestamp (int hour_offset = 0,                   [Function]
        int min_offset = 0,
        time_t *seconds = 0)
```

26.14 Pull functions

`void* pull_request (void *v)` [Thread function]

```
int pull_client_func (Scan_Parse_Parameter_Type &param,      [Function]
        string distinguished_name_str)
```

26.15 Other functions

`void lock_cerr_mutex (void)` [Functions]
`void unlock_cerr_mutex (void)`

`int submit_mysql_query (string query,` [Function]
`MYSQL_RES *&result,`
`MYSQL *mysql_ptr,`
`unsigned int *row_ctr = 0,`
`unsigned int *field_ctr = 0,`
`long *affected_rows = 0,`
`string thread_ctr_str = "")`

`int init_gw_code_map (void)` [Function]
Defined in '`gwrdifpk-1.0/src/rspercds`'. This function initializes the global variable `map<int, string> gw_code_map`. See Section 13.2 [Global variables], page 102, and Section 13.1.1 [Response and error codes], page 101.

`string gwstrerror (int code, bool suppress_if_unknown)` [Function]
Returns a `string` with a human-readable message for the response or error code *code*. The messages are stored in the global variable `map<int, string> gw_code_map`. See Section 13.2 [Global variables], page 102.

If *code* is unknown, a message to this effect is returned instead, unless *suppress_if_unknown* = `true`, in which case the empty string is returned.

This function is defined in '`gwrdifpk-1.0/src/rspercds`'. See Section 13.1.1 [Response and error codes], page 101.

`int set_debug_level (bool &DEBUG,` [Function]
`int turn_on_value = 0,`
`int turn_off_value = 0)`

`int hexl_encode (const char *buffer,` [Function]
`unsigned int buffer_size,`
`string &result,`
`int delimiter = -1,`
`int delimiter_1 = -1)`

`int hexl_decode (string &source,` [Function]
`string &dest,`
`unsigned int &dest_length)`

`int check_irods_server (int &pid, int thread_ctr = -1)` [Function]

`int set_password (string filename,` [Function]
`char *&password,`
`size_t password_length,`
`string default_filename,`
`char *passphrase = 0,`
`string directory = "",`
`string thread_str = "")`

```
int write_to_fifo (char *buffer,                                    [Function]
        size_t buffer_length,
        int fd,
        bool write_string_length = true,
        bool append_newline = true,
        string thread_str = "")
```

27 handlesystem and handlesystem_standalone Databases

The '`handlesystem`' database is used by a handle server configured to use an "official" handle prefix assigned by CNRI, while the '`handlesystem_standalone`' database is used one configured to ask as a "standalone" handle server using "internal" prefixes outside the domain defined by CNRI. See Section 1.2 [Handles], page 2.

27.1 nas

```
na          varchar(255) not null primary key
```

27.2 handles

```
handle      varchar(255) not null
idx         int4 not null
type        blob
data        blob
ttl_type    int2
ttl         int4
timestamp
            int4
refs        blob
admin_read
            bool
admin_write
            bool
pub_read    bool
pub_write
            bool
created     timestamp default 0
last_modified
            timestamp default 0
created_by_user_id
            int
irods_object_id
            bigint unsigned not null default 0
handle_id
            bigint unsigned not null default 0
handle_value_id
            bigint unsigned not null default 0
deleted     boolean not null default false
PRIMARY KEY(handle, idx)
```

27.3 admin_data

```
handle     varchar(255) not null

data       blob
```

27.4 pid_counters

```
prefix     varchar(16) primary key not null

pid_counter
           bigint unsigned not null
```

28 gwirdsif Database

28.1 Users

`user_id` `int primary key`

`username` `varchar(128) unique not null`

`irods_password_encrypted`
`varchar(2048)` $(2048 \equiv 2^{11})$

`irods_password_encrypted_timestamp`
`timestamp default 0`

`Distinguished_Name`
`varchar(256)`

`irods_homedir`
`varchar(128)`

`irods_zone`
`varchar(128)`

`irods_default_resource`
`varchar(128)`

`handle_username varchar(128)`
`default ''`

`handle_password_encrypted`
`varchar(32) default ''`

`default_institute_id`
`int references Institutes(institute_id)`

`default_prefix_id`
`int references Prefixes(prefix_id)`

28.2 User_Info

The view `User_Info` combines columns from several tables in order to be able to present information about users in a convenient way. It is defined by the following SQL query:

```
create view User_Info as select
   U.user_id, U.username,
   C.certificate_id, C.serialNumber, C.commonName, C.organization,
   C.organizationalUnitName,
   P.superuser, P.delegate, P.show_user_info, P.show_groups,
   P.show_certificates, P.show_distinguished_names,
   P.show_privileges,
   G.gpg_key_pair_id, G.fingerprint,
   G.created as 'GPG key pair created',
   G.last_modified as 'GPG key pair last modified'
from Users as U, Certificates as C, Privileges as P, GPG_Key_Pairs as G
```

```
where U.user_id = C.user_id and U.user_id = P.user_id
   and U.user_id = G.user_id
order by U.user_id, G.gpg_key_pair_id;
```

Example:

```
select * from User_Info where user_id = 1\G

⇒
*************************** 1. row ***************************
                   user_id: 1
                  username: lfinsto
             certificate_id: 2
               serialNumber: 2
                 commonName: Laurence Finston
               organization: GWDG
    organizationalUnitName: gwrdifpk
                 superuser: 1
                  delegate: 1
            show_user_info: 1
               show_groups: 1
          show_certificates: 1
  show_distinguished_names: 1
           show_privileges: 1
           gpg_key_pair_id: 1
               fingerprint: 41E4286D5DED32B80956D5429CBFF6B2CA0E93A2
      GPG key pair created: 2014-01-16 13:32:42
GPG key pair last modified: 0000-00-00 00:00:00
1 row in set (0.00 sec)
```

28.3 Certificates

The `Certificates` table stores information from *X.509 certificates*. Its columns correspond to the data members of `class X509_Certificate_Type`. See Section 5.3 [X.509 Certificates], page 62, and Chapter 20 [X.509 Certificate Types], page 134.

`certificate_id`
: `int primary key`

`user_id` `int references Users(user_id)`

`issuer_cert_id`
: `int references Certificates(certificate_id)`

`is_ca` `boolean not null` (Is certification authority)

`is_proxy` `boolean not null` (Is proxy certificate)

The following columns have names from the X.509 specification:

`serialNumber`
: `bigint unsigned`

```
organization
          varchar(64)

organizationalUnitName
          varchar(64)

commonName
          varchar(64)

countryName
          char(2)

localityName
          varchar(64)

stateOrProvinceName
          varchar(64)

Validity_notBefore
          datetime

Validity_notAfter
          datetime
```

28.4 Groups database tables and views

See also Chapter 17 [class Group_Type], page 121.

28.4.1 Groups

```
group_id  int primary key unique not null

creator_id
          int not null references Users(user_id)

name      varchar(64) unique not null

created   datetime
```

28.4.2 Groups_Users

```
group_id  int not null references Groups(group_id)

user_id   int not null references Users(user_id)

add_user_priv
          boolean not null default 0

delete_user_priv
          boolean not null default 0

delete_group_priv
          boolean not null default 0
```

28.4.3 Group_Info (view)

The view `Group_Info` combines columns from the tables `Groups`, `Users` and `Groups_Users` in order to be able to present information about groups in a convenient way. It is defined by the following SQL query:

```
create view Group_Info as select
   GU.group_id, G.name as group_name,
   GU.user_id, U.username as 'username',
   GU.add_user_priv, GU.delete_user_priv, GU.delete_group_priv,
   G.creator_id, UU.username as creator_name, G.created
from Groups as G, Users as U, Groups_Users as GU, Users as UU
where G.group_id = GU.group_id and U.user_id = GU.user_id
and G.creator_id = UU.user_id
order by G.group_id, GU.user_id;
```

Example:

```
select * from Group_Info where group_id = 1\G

⇒

*************************** 1. row ***************************
         group_id: 1
       group_name: test_group_0
          user_id: 1
         username: lfinsto
    add_user_priv: 1
 delete_user_priv: 1
delete_group_priv: 1
       creator_id: 1
     creator_name: lfinsto
          created: 2013-06-05 14:05:54
*************************** 2. row ***************************
         group_id: 1
       group_name: test_group_0
          user_id: 2
         username: jdoe
    add_user_priv: 0
 delete_user_priv: 0
delete_group_priv: 0
       creator_id: 1
     creator_name: lfinsto
          created: 2013-06-05 14:05:54
2 rows in set (0.01 sec)
```

28.5 Institutes

```
institute_id
          int primary key
```

```
contact    int references Users(user_id)

abbreviation
           char(4) unique not null

name       varchar(128) unique not null

enabled    boolean not null default 1
```

28.6 Prefixes

```
prefix_id
           int primary key

prefix     varchar(16) unique not null

enabled    boolean not null default 1
```

28.7 Users_Prefixes

```
user_id    int references Users(user_id),

prefix_id
           int references Prefixes(prefix_id)
```

28.8 Privileges

See also Section 10.1 [Privileges], page 98.

```
user_id    int primary key unique not null references Users(user_id)

superuser
           boolean not null default 0

delegate   boolean not null default 0

add_groups
           boolean not null default 0

delete_groups
           boolean not null default 0

delete_handles
           boolean not null default 0

delete_handle_values
           boolean not null default 0

delete_hs_admin_handle_values
           boolean not null default 0

delete_last_hs_admin_handle_value
           boolean not null default 0

undelete_handle_values
           boolean not null default 0
```

```
show_user_info
          boolean not null default 0

show_groups
          boolean not null default 0

show_certificates
          boolean not null default 0

show_distinguished_names
          boolean not null default 0

show_privileges
          boolean not null default 0
```

28.9 Public_Keys

```
user_id   int primary key references Users(user_id),

key_name  varchar(256) not null,

key_id    int unsigned not null
```

28.10 Irods_Objects database tables

See also Chapter 18 [iRODS Types], page 122.

28.10.1 Irods_Objects

See also Section 18.1 [class Irods_Object_Type], page 122.

```
irods_object_id
          bigint unsigned primary key not null

user_id   int not null references Users(user_id)

irods_server_id
          int unsigned not null references Irods_Servers(irods_server_id)

irods_object_path
          varchar(1024) not null default ''

marked_for_deletion_from_archive
          boolean not null default 0

deleted_from_archive
          boolean not null default 0

delete_from_archive_timestamp
          timestamp not null default 0

marked_for_deletion_from_gwirdsif_db
          boolean not null default 0

delete_from_gwirdsif_db_timestamp
          timestamp not null default 0
```

```
created    timestamp not null default 0

last_modified
           timestamp not null default 0

dublin_core_metadata_id
           bigint unsigned not null default 0 references Dublin_Core_
           Metadata(dublin_core_metadata_id)

dublin_core_metadata_irods_object_id
           bigint unsigned not null default 0 references Irods_Objects(irods_
           object_id)

irods_object_ref_id
           bigint unsigned not null default 0 references Irods_Objects(irods_
           object_id)

encrypted
           boolean not null default 0

signed_gpg
           boolean not null default 0

detached_signature_irods_object_id
           bigint unsigned not null default 0 references Irods_Objects(irods_
           object_id)

gpg_key_pair_id_encrypt
           int unsigned not null default 0 references GPG_Key_Pairs(gpg_key_
           pair_id)

gpg_key_pair_id_sign
           int unsigned not null default 0 references GPG_Key_Pairs(gpg_key_
           pair_id)

gpg_key_fingerprint_encrypt
           varchar(64) not null default ''

gpg_key_fingerprint_sign
           varchar(64) not null default ''

compressed_tar_file
           boolean not null default 0

compressed_gzip
           boolean not null default 0

compressed_bzip2
           boolean not null default 0
```

28.10.2 Irods_AVUs

See also Section 18.2 [class Irods_AVU_Type], page 125.

```
irods_avu_id
           bigint unsigned primary key not null
```

```
irods_object_id
          bigint unsigned references Irods_Objects(irods_object_id)
attribute
          varchar(512)
value     varchar(512)
units     varchar(16)
time_set  timestamp default 0
```

28.10.3 Irods_Servers

```
irods_server_id
          int unsigned primary key not null
irods_server_name
          varchar(1024) not null default ''
irods_server_ip_address
          varchar(1024)
irods_server_port
          int unsigned not null
irods_server_admin
          varchar(1024)
```

28.10.4 Irods_Objects_Handles

```
irods_object_id
          bigint unsigned references Irods_Objects(irods_object_id)
handle_id
          bigint unsigned references handlesystem_standalone.handes(handle_
          id)
```

28.10.5 Irods_Objects_Dublin_Core_Metadata

Association table. This table makes it possible to associate multiple sets of Dublin Core metadata to a given iRODS object. It also makes it possible to associate multiple iRODS objects to a single set of Dublin Core metadata, if this should ever be desired. At the present time, the author doesn't see any need to do this.

Please note that, strictly speaking, this table is not really needed, because the `irods_object_ref_id` field in the `Dublin_Core_Metadata` table could be used to determine which rows in `Dublin_Core_Metadata` correspond to ones in the `Irods_Objects` table. However, the `Irods_Objects_Dublin_Core_Metadata` simplifies this task.

```
irods_object_id
          bigint unsigned not null default 0 references Irods_Objects(irods_
          object_id)
dublin_core_metadata_id
          bigint unsigned not null default 0 references Dublin_Core_
          Metadata(dublin_core_metadata_id)
```

28.10.6 Irods_Info (view)

The view `Irods_Info` combines columns from the tables `Irods_Objects` and `Irods_AVUs` in order to be display information about iRODS objects together with information about all of their AVUs. It is defined by the following SQL query:

```
create view Irods_Info as
select IO.irods_object_id, IA.irods_avu_id, IO.user_id,
IO.irods_object_path, IA.attribute,
IA.value, IA.units, IA.time_set as 'AVU timeset',
IO.created as 'irods object created',
IO.last_modified as 'irods object last modified',
IO.marked_for_deletion_from_archive as
'irods_object_marked_for_deletion_from_archive',
IO.marked_for_deletion_from_gwirdsif_db as
'irods_object_marked_for_deletion_from_gwirdsif_db',
IO.deleted_from_archive as 'irods_object_deleted_from_archive',
IO.delete_from_archive_timestamp as
'irods_object_delete_from_archive_timestamp',
IO.delete_from_gwirdsif_db_timestamp as
'irods_object_delete_from_gwirdsif_db_timestamp'
from Irods_Objects as IO, Irods_AVUs as IA
where IO.irods_object_id = IA.irods_object_id
and IO.irods_object_id > 0 and IA.irods_avu_id > 0
order by IO.irods_object_id, IA.irods_avu_id;
```

Example:

```
select * from Irods_Info where irods_object_id = 1\G
```

⇒

```
*************************** 1. row ***************************
                    irods_object_id: 1
                       irods_avu_id: 1
                            user_id: 1
                   irods_object_path: /tempZone/home/lfinsto/abc.txt
                          attribute: PID
                              value: 12345/00001
                              units:
                         AVU timeset: 2013-10-24 11:15:48
                irods object created: 2013-10-24 11:15:48
           irods object last modified: 0000-00-00 00:00:00
    irods_object_marked_for_deletion_from_archive: 0
irods_object_marked_for_deletion_from_gwirdsif_db: 0
               irods_object_deleted_from_archive: 0
       irods_object_delete_from_archive_timestamp: 0000-00-00 00:00:00
   irods_object_delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
*************************** 2. row ***************************
                    irods_object_id: 1
                       irods_avu_id: 2
                            user_id: 1
                   irods_object_path: /tempZone/home/lfinsto/abc.txt
                          attribute: DC_METADATA_PID
                              value: 12345/00002
```

```
                                          units:
                                    AVU timeset: 2013-10-24 11:15:54
                                irods object created: 2013-10-24 11:15:48
                            irods object last modified: 0000-00-00 00:00:00
        irods_object_marked_for_deletion_from_archive: 0
    irods_object_marked_for_deletion_from_gwirdsif_db: 0
                    irods_object_deleted_from_archive: 0
           irods_object_delete_from_archive_timestamp: 0000-00-00 00:00:00
       irods_object_delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
    *************************** 3. row ***************************
                                   irods_object_id: 1
                                      irods_avu_id: 8
                                           user_id: 1
                                  irods_object_path: /tempZone/home/lfinsto/abc.txt
                                          attribute: DC_METADATA_IRODS_OBJECT_REF
                                              value: \
       /tempZone/home/lfinsto/metadata_sample_1.xml
                                          units:
                                    AVU timeset: 2013-10-24 11:15:54
                                irods object created: 2013-10-24 11:15:48
                            irods object last modified: 0000-00-00 00:00:00
        irods_object_marked_for_deletion_from_archive: 0
    irods_object_marked_for_deletion_from_gwirdsif_db: 0
                    irods_object_deleted_from_archive: 0
           irods_object_delete_from_archive_timestamp: 0000-00-00 00:00:00
       irods_object_delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
    *************************** 4. row ***************************
                                   irods_object_id: 1
                                      irods_avu_id: 9
                                           user_id: 1
                                  irods_object_path: /tempZone/home/lfinsto/abc.txt
                                          attribute: DC_METADATA_IRODS_OBJECT_PID
                                              value: 12345/00003
                                          units:
                                    AVU timeset: 2013-10-24 11:15:54
                                irods object created: 2013-10-24 11:15:48
                            irods object last modified: 0000-00-00 00:00:00
        irods_object_marked_for_deletion_from_archive: 0
    irods_object_marked_for_deletion_from_gwirdsif_db: 0
                    irods_object_deleted_from_archive: 0
           irods_object_delete_from_archive_timestamp: 0000-00-00 00:00:00
       irods_object_delete_from_gwirdsif_db_timestamp: 0000-00-00 00:00:00
    4 rows in set (0.00 sec)
```

28.11 Session_Data

```
session_id
          varchar(256) not null

user_id   int not null references Users(user_id)

effective_user_id
          int references Users(user_id)

user_name
          varchar(128) references Users(user_name)
```

```
effective_user_name
          varchar(128) references Users(user_name)
timestamp
          timestamp default 0
```

28.12 TANs

```
user_id   int default 0 references Users(user_id)
TAN       varchar(64) primary key
expiration
          timestamp default 0
```

28.13 Dublin Core database tables

See also Chapter 21 [Dublin Core Metadata Types], page 138.

28.13.1 Dublin_Core_Metadata

```
dublin_core_metadata_id
          bigint unsigned primary key not null
user_id   int not null references Users(user_id)
irods_server_id
          int unsigned references Irods_Servers(irods_server_id)
irods_object_path
          varchar(1024)
handle_id
          bigint references handlesystem_standalone(handle_id)
dc_metadata_irods_object_path
          varchar(1024)
created   datetime not null default 0
last_modified
          datetime not null default 0
marked_for_deletion
          boolean not null default false
delete_file
          tinyint not null default 0
delete_from_database_timestamp
          timestamp default 0
irods_object_ref_id
          bigint unsigned not null default 0 references Irods_Objects(irods_
          object_id)
irods_object_self_id
          bigint unsigned not null default 0 references Irods_Objects(irods_
          object_id)
```

28.13.2 Dublin_Core_Metadata_Sub

```
dublin_core_metadata_sub_id
          bigint unsigned primary key not null

dublin_core_metadata_id
          bigint unsigned not null references Dublin_Core_Metadata(dublin_
          core_metadata_id)

dublin_core_element_id
          int unsigned not null default 0 references Dublin_Core_
          Elements(dublin_core_element_id)

dublin_core_term_id
          int unsigned not null default 0 references Dublin_Core_
          Terms(dublin_core_term_id)

value     varchar(1024) not null default ''
```

28.13.3 Dublin_Core_Elements

```
dublin_core_element_id
          int unsigned primary key not null

element_name
          varchar(32) not null
```

28.13.4 Dublin_Core_Terms

```
dublin_core_term_id
          int unsigned primary key not null

term_name
          varchar(64) not null
```

28.13.5 Dublin_Core_Qualifiers

```
dublin_core_qualifier_id
          bigint unsigned primary key not null

dublin_core_metadata_id
          bigint unsigned references Dublin_Core_Metadata(dublin_core_
          metadata_id)

dublin_core_element_id
          bigint unsigned references Dublin_Core_Elements(dublin_core_
          element_id)

dublin_core_term_id
          bigint unsigned references Dublin_Core_Terms(dublin_core_term_id)

value     varchar(512) not null default ''
```

28.13.6 Dublin_Core_Attributes

```
dublin_core_metadata_id
          bigint unsigned not null references Dublin_Core_Metadata(dublin_
          core_metadata_id)

dublin_core_metadata_sub_id
          bigint unsigned not null references Dublin_Core_Metadata_
          Sub(dublin_core_metadata_sub_id)

attribute
          varchar(128)

value     varchar(256)
```

28.14 GPG_Key_Pair database tables

28.14.1 GPG_Key_Pairs

```
gpg_key_pair_id
          int unsigned primary key not null default 0

user_id   int not null default 0 references Users(user_id)

fingerprint
          varchar(64) not null default ''

public_key
          blob not null

created   datetime not null default 0

last_modified
          datetime not null default 0
```

28.14.2 Users_GPG_Key_Pairs

```
user_id   int not null default 0 references Users(user_id)

gpg_key_pair_id
          int unsigned not null default 0 references GPG_Key_Pairs(gpg_key_
          pair_id)
```

28.15 Pull Request database table

```
pull_request_id
          int primary key

user_id   int not null default 0 references Users(user_id)

server_hostname
          varchar(128) not null default ''

server_ip_address
          varchar(64) not null default ''
```

```
client_hostname
          varchar(128) not null default ''

client_ip_address
          varchar(64) not null default ''

client_port
          int unsigned not null default 0

pull_interval
          int not null default 0

latest_pull
          datetime not null default 0

created   datetime not null default 0

last_modified
          datetime not null default 0
```

29 gwirdcli Database

The client-side database `gwirdcli` contains tables for storing data sent by the server program `gwirdsif` to the client program `gwirdcli`. The tables are based on ones defined in the `gwirdsif` and `handlesystem` or (`handlesystem_standalone`) databases. See Chapter 28 [gwirdsif Database], page 160, and Chapter 27 [handlesystem and handlesystem_standalone Databases], page 158.

29.1 Users

See Section 28.1 [Users database table (gwirdsif)], page 160.

29.2 handles

See Section 27.2 [handles database table], page 158.

29.3 nas

See Section 27.1 [nas database table], page 158.

29.4 Privileges_Gwirdcli

```
user_id    int primary key unique not null references Users(user_id)
superuser
           boolean not null default 0
delegate   boolean not null default 0
add_groups
           boolean not null default 0
delete_groups
           boolean not null default 0
show_user_info
           boolean not null default 0
show_groups
           boolean not null default 0
show_certificates
           boolean not null default 0
show_distinguished_names
           boolean not null default 0
show_privileges
           boolean not null default 0
pull_response_self
           boolean not null default 0
pull_response_group
           boolean not null default 0
```

```
pull_response_all
          boolean not null default 0
```

29.5 Dublin Core database tables

The `gwirdcli` database contains the following tables:

- `Dublin_Core_Metadata`
- `Dublin_Core_Metadata_Sub`
- `Dublin_Core_Attributes`
- `Dublin_Core_Elements`
- `Dublin_Core_Qualifiers`
- `Dublin_Core_Terms`

The table definitions are identical to the ones in the `gwirdsif` database, with one exception: the `Dublin_Core_Metadata` table contains one additional column:

```
retrieved_from_server_timestamp timestamp default 0
```

See Section 28.13 [Dublin Core database tables (gwirdsif)], page 170.

29.6 GPG_Key_Pair database tables

The definitions for the database tables `gwirdcli.GPG_Key_Pairs` and `gwirdcli.Users_GPG_Key_Pairs` are identical to those for the corresponding tables in the `gwirdsif` database.
See Section 28.14 [GPG_Key_Pair database tables (gwirdsif)], page 172.

29.7 Pull Response database tables

29.7.1 Pull_Servers

```
pull_server_id
          int primary key

server_hostname
          varchar(128) not null default ''

server_ip_address
          varchar(64) not null default ''

server_distinguished_name
          varchar(256) not null default ''

created   datetime not null default 0

last_modified
          datetime not null default 0
```

29.7.2 Pull_Responses

```
pull_response_id
          int primary key

user_id   int not null default 0 references Users(user_id)
```

```
server_hostname
          varchar(128) not null default ''

server_ip_address
          varchar(64) not null default ''

client_hostname
          varchar(128) not null default ''

client_ip_address
          varchar(64) not null default ''

pull_interval
          int not null default 0

latest_pull
          datetime not null default 0

created   datetime not null default 0

last_modified
          datetime not null default 0
```

29.7.3 Pull_Paths

```
pull_path_id
          int primary key

pull_response_id
          int references Pull_Responses(pull_response_id)

owner_id  int not null default 0 references Users(user_id)

local_path
          varchar(512) not null default ''

remote_path
          varchar(512) not null default ''

checksum_sha224
          varchar(64) not null default ''

created   datetime not null default 0

last_modified
          datetime not null default 0
```

30 Profiling and testing

31 Web application gwrdwbap

The web application `gwrdwbap` exists, but as of 2013.10.31., it is not functional and is not being developed. It may be desirable to continue development on it at some future time.

Source files:

‘`gwrdwbap.web`’

‘`ckietype.web`’

‘`utlwbfcs.web`’

32 Auxiliary programs

32.1 Generate PIDs (genpids)

`genpids` [`OPTION` ...] [`TYPE` `VALUE` ...] [Command]

Create one or more PIDs (handles).

`genpids` is a "wrapper" program for the function `generate_pids`. See Section 26.8 [Generating PIDs], page 153. Entries are created in the `handlesystem.handles` or `handlesystem_standalone.handles` database table and the PIDs are printed to standard output.

Options:

`--pid` `STRING`

`STRING` is used for the PID. If this name (together with the prefix) conflicts with an existing handle, an error message is issued and `genpids` exists unsuccessfully.

`STRING` may or may not contain a slash character ('`/`'). If it does, the portion of it preceding the slash is taken to be the prefix and the portion following it is the *handle suffix*

`--prefix` `STRING`

`STRING` is used as the prefix. Otherwise, the default prefix '`00001`' is used. If the system is not configured to use the prefix `STRING`, an error message is issued and `genpids` exists unsuccessfully.

`--suffix-additional` `STRING`

`STRING` is an additional, user-defined suffix. It is appended to the end of the PID, i.e., following the *handle suffix* or *local name*, and preceded by a hyphen.

Please note: This option is ignored if the '`--pid`' option is also specified! In this case, a warning is issued. Any user-defined suffix may be included in the `STRING` argument to the '`--pid`' option, so there's no need to use the two options together.

`--institute` `STRING`

`STRING`, followed by hyphen, is inserted after the prefix and before the handle suffix or local name. `STRING` must consist of exactly 4 characters.

Please note: This option is ignored if the '`--pid`' option is also specified! In this case, a warning is issued. Any string for identifying an institute may be included in the `STRING` argument to the '`--pid`' option, so there's no need to use the two options together.

`--suppress-prompt`

Suppress the prompt for the MySQL password. This option may be used when passing the latter to `genpids` via a pipe.

`--number-of-pids` `INTEGER`

Create `INTEGER` PIDs (handles).

Please note: This option is ignored if the '`--pid`' option is also specified! In this case a warning is issued. Specifying a PID using the '`--pid`' option implies that only one PID should be created.

`--fifo-pathname PATH`
: The function `generate_pids` will write the PIDs to the FIFO `PATH`. This option will generally not be needed. See Section 26.8 [Generating PIDs], page 153.

`--username STRING`
: Create the PID or PIDs for user `STRING`. Otherwise, they are created for the user who invoked the program. If there is no entry for user `STRING` in the '`gwirdsif`' database, `genpids` will issue an error message exit unsuccessfully.

`--user-id INTEGER`
: Specify the user ID. If this option is used, the database isn't queried for the user ID to be stored in the `created_by_user_id` field of the handle or handles. **Please note**: This is not bullet-proof! `INTEGER` will *always* be used, even if it doesn't correspond with the argument to the '`--username`' option or the username of the user who invoked the program. It will even be used if there is no entry in the '`gwirdsif`' database for a user with this ID!

`--query-database`
: After the PID or PIDs are created, an SQL `select` statement is passed to the command-line program `mysql` and the latter's output is printed to standard output.

`--help` `genpids` issues a help message and exits.

`--version`
: `genpids` prints version information to standard output and exits.

Additional non-option arguments are "type-value pairs", where `TYPE` and `VALUE` are both strings. The '`type`' and `data` fields of the handle or handles are set to `TYPE` and `VALUE`, respectively. If `TYPE` is a "known" type, i.e., there are entries for it `Handle_Value_Type::idx_type_map` and `Handle_Value_Type::idx_type_map`, the `idx` field is set to the corresponding value, or a value is found according to rules defined within the function `Handle_Type::add_values`. See Section 19.1.2 [Handle_Type Member Functions], page 127.

Examples:

```
echo "<MySQL password>" | genpids

⇒

00001/00012

echo "<MySQL password>" | genpids --suppress --prefix 12345
```

```
⇒

12345/00013

echo "<MySQL password>" | genpids --suppress --suffix XXX

⇒

00001/00014-XXX

echo "<MySQL password>" | genpids --suppress --institute ZZZZ

⇒

00001/ZZZZ-00015

echo "<MySQL password>" | genpids --suppress --pid "12345/12AB"

⇒

12345/12AB

echo "<MySQL password>" | genpids --suppress --number-of-pids 3

⇒

00001/00016
00001/00017
00001/00018

echo "<MySQL password>" | genpids --suppress --query \
   IRODS_OBJECT_TYPE "abc.txt"

⇒

00001/00019
*************************** 1. row ***************************
                        handle: 00001/00019
                           idx: 300
                          type: HS_ADMIN
                          data: [binary]
                      ttl_type: 0
                           ttl: 86400
                     timestamp: 1383146594
                          refs:
                    admin_read: 1
                   admin_write: 1
```

```
                        pub_read: 1
                       pub_write: 0
                         created: 2013-10-30 16:23:14
                   last_modified: 0000-00-00 00:00:00
               created_by_user_id: 1
                  irods_object_id: 0
                        handle_id: 124
                  handle_value_id: 221
              marked_for_deletion: 0
delete_from_database_timestamp: 0000-00-00 00:00:00
*************************** 2. row ***************************
                         handle: 00001/00019
                            idx: 601
                           type: IRODS_OBJECT_TYPE
                           data: abc.txt
                       ttl_type: 0
                            ttl: 86400
                      timestamp: 1383146594
                           refs:
                     admin_read: 1
                    admin_write: 1
                       pub_read: 1
                       pub_write: 0
                         created: 2013-10-30 16:23:14
                   last_modified: 0000-00-00 00:00:00
               created_by_user_id: 1
                  irods_object_id: 0
                        handle_id: 124
                  handle_value_id: 222
              marked_for_deletion: 0
delete_from_database_timestamp: 0000-00-00 00:00:00
```

32.2 Generate TANs (gentans)

The program `gentans` is not currently in use, because `gwrdifpk` has not been set up to use TANs. It exists because it may be desirable to implement this feature in the future.

32.3 Process scanner and parser input files (prbsnflx)

`prbsnflx` is a simple program for processing the output of the `ctangle` program, so that it may be passed to Flex or GNU Bison as input. `prbsnflx` is a "wrapper" for a Flex scanner. It removes comments and preprocessor commands which are valid input for C and C++, but not for Flex or GNU Bison.

`prbsnflx` is defined in '`gwrdifpk-1.0/src/prbsnflx.l++`'.

Users will not normally need to invoke `prbsnflx`. It is invoked in the '`make`' rules in '`gwrdifpk-1.0/src/Makefile.am`' for generating the Flex and Bison input files. See Section 26.6 [Scanning and parsing], page 152.

32.4 Generate Passwords or Passphrases (optpsgen)

`optpsgen` generates one or more passwords or passphrases from randomly chosen characters. Options control the type of characters used, whether whitespace may be included, and other characteristics of the passwords or passphrases. Additionally, checksums may be output, using one of several checksum functions, i.e., md5, sha1, sha224, sha256, sha384, or sha512.

The files '`optpsgen.web`' and '`optpsgsb.web`' contain the source code.

32.4.1 Options

Options:

'`--help`' Outputs a message explaining usage and listing these options and exits.

'`--debug-level INTEGER`'
: If INTEGER > 0, debugging information is output. Currently, there is only one "debugging level", i.e., the magnitude of INTEGER otherwise makes no difference.

'`--input-filename FILENAME`'
: For testing or debugging. The file FILENAME will be used instead of '`/dev/urandom`' (the default) or '`/dev/random`' (when the `--extra-random` option is used).

'`--alpha`'
'`--alphanum, --alnum`'
'`--graph`'
'`--printable`'
: These options determine what types of characters may appear in the passwords or passphrases, i.e., alphabetical, alphanumeric, "graphical", or printable, respectively. They correspond to the C functions `isalpha`, `isalnum`, `isgraph` and `isprint`. That is, the characters allowed depend on the current locale.

'`--blank`'
'`--space`'
'`--no-tabs`'
: Intended for use with the options '`--alpha`', '`--alphanum`' and '`--alnum`'. If '`--space`' is used, then whitespace characters are also allowed, while if '`--blank`' is used, only the actual space character (ASCII 32) and the tab character (ASCII 9) are allowed. '`--no-tabs`' causes tabs to be suppressed. '`--blank --no-tabs`' therefore has the effect of allowing the actual space character, but no other whitespace characters.

  These options have no effect if used together with '`--printable`' or '`--graph`', because '`--graph`' is equivalent to '`--printable`' minus whitespace. If desired, space characters may be included in passwords or passphrases generated using '`--graph`' (or '`--printable`') by using the '`--max-block-size`' option (see below).

'`--length INTEGER`'
: Specifies the length of the password or passphrase. Default is 8 characters.

‘`--extra-random`’
: Use ‘`/dev/random`’ instead of ‘`/dev/urandom`’ as the source of characters. This improves the quality of the “randomness”, but may cause the program to block, if not enough entropy is present in the system. A message to this effect is issued.

‘`--min-block-size [INTEGER]`’
‘`--max-block-size [INTEGER] (default 8)`’
: Set the minimum and/or maximum size of “blocks” of non-whitespace characters. The argument is optional. The default for the minimum block size is 4, while that for the maximum is 8. Please note that these defaults only apply *if the corresponding option is specified*. Otherwise, there is no minimum or maximum block size.

  These options have no effect if used with ‘`--graph`’, because the latter excludes whitespace entirely.

‘`--no-start-space`’
‘`--no-end-space`’
: Prevent whitespace from occurring at the beginning or end of the password, respectively.

‘`--delimiters [ARG]`’
: The password or passphrase will be output with a “delimiter” at the beginning and end. ARG is optional. If not used, ‘’ will be used on both sides. If present, it should be a character or string. If it is a character or a string containing only one character, this character will be used on both sides. If it is a string containing more than one character, the first character will be used at the beginning and the second at the end. If there are more than two characters, the remaining ones are ignored.

  Delimiters can be useful if whitespace may appear at the beginning and/or end of the password or passphrase.

‘`--exclude-chars STRING`’
: STRING argument required. It is a list of characters which should *not* appear in the password or passphrase.

‘`--checksum [ARG]`’
: Output a checksum for the generated password or passphrase. If ARG is not present, ‘`sha1sum`’ is used to generate the checksum. Valid values for ARG are ‘`md5`’, ‘`sha1`’, ‘`sha224`’, ‘`sha224`’, ‘`sha384`’ or ‘`sha512`’.

‘`--iterations INTEGER`’
: The number of passwords or passphrases (and optionally checksums) to generate.

32.4.2 Global Variables

`extern const unsigned int DEFAULT_PASSWD_LEN = 8` [Constant]
: Default length of password or passphrase.

`unsigned int passwd_len` [Variable]
: Set to `DEFAULT_PASSWD_LEN` at the beginning of `main`.

`int debug_level = 0` [Variable]

Set by the '`--debug`' option. See Section 32.4.1 [optpsgen Options], page 183.

```
extern const unsigned int ALPHA_TYPE = 1                    [Constants]
extern const unsigned int ALPHANUM_TYPE = 2
extern const unsigned int GRAPH_TYPE = 4
extern const unsigned int PRINTABLE_TYPE = 8
extern const unsigned int BLANK_TYPE = 16
extern const unsigned int SPACE_TYPE = 32
extern const unsigned int NO_TABS_TYPE = 64
extern const unsigned int NO_START_SPACE_TYPE = 128
extern const unsigned int NO_END_SPACE_TYPE = 256
```

Constants for controlling what characters may appear in the password or passphrase.

```
extern const unsigned int MD5_TYPE = 1                      [Constants]
extern const unsigned int SHA1_TYPE = 2
extern const unsigned int SHA224_TYPE = 3
extern const unsigned int SHA256_TYPE = 4
extern const unsigned int SHA384_TYPE = 5
extern const unsigned int SHA512_TYPE = 6
```

```
extern const int DEFAULT_MIN_BLOCK_SIZE = 4                 [Constants]
extern const int DEFAULT_MAX_BLOCK_SIZE = 8
int min_block_size = 0                                      [Variables]
int max_block_size = 0
unsigned int block_ctr = 0
```

`bool extra_random = false` [Variable]

```
string delim_start                                          [Variables]
string delim_end
```

`string in_filename` [Variable]

`vector<char> exclude_char_vector` [Variable]

`unsigned int checksum_type = 0` [Variable]

`unsigned int iterations = 1` [Variable]

32.4.3 Functions

`int main` (*int* `argc`, *char* `**argv`) [Function]

`int handle_options` (*int* `argc`, *char* `**argv`) [Function]

32.5 Set up databases (setupdbs)

`setupdbs` can be used to set up the databases needed by `gwrdifpk`. Invoking it can be as simple as '`./setupds`', but it has various options which can be used to control its behavior, as described in the following section.

32.5.1 Invoking

`setupds` uses the `getopt_long_only` function from the GNU C library to parse its command-line arguments. This implies that the options to `setupds` may be specified using two hyphens, as in the list below, or with a single hyphen. In addition, any option may be abbreviated, as long as the abbreviation is unambiguous. For example, '`--version`' may be specified as '`-version`', '`--ver`' or even '`-v`', since there (currently) are no other options whose names begin with "v". On the other hand, the option '`--gwirdsif-database-name`' may be abbreviated to '`--gwirds`' but not to to '`--gwird`', because the option '`--gwirdcli-database-name`' also begins with this sequence of characters, making '`--gwird`' ambiguous. See Section "Getopt Long Options" in *The GNU C Library Reference Manual*.

Options:

`--drop`
: If any of the databases exist, they are dropped before being recreated. If any of the databases exist and this option is *not* specified, `setupdbs` exits with exit status 2.

`--admin <USERNAME>`
`--user <USERNAME>`
: For '`--admin`', the user specified by `<USERNAME>` is made an administrator, while for '`--user`', he or she is made a normal user without administrative privileges. The user specified as an argument to the first occurrence of the '`--admin`' option is given the user ID 1. This ID is only ever given to an "admin", i.e., if the '`--admin`' option is not used, but the '`--user`' option is, the user specified by the first occurrence of the latter is given user ID 2. These options may each be used multiple times.

`--group <STRING>`
: Group `<STRING>` will be created. This option may be used multiple times.

`--handles-database-name <STRING>`
: The default value depends on the value of *standalone_handle*, which is `true` by default and set to `false` by the '`--no-standalone`' option (see above). For a registered handle service, the default name is `handlesystem`, while for a standalone handle service, it is `handlesystem_standalone`. For testing only. See below.

`--server-database-name <STRING>`
`--gwirdsif-database-name <STRING>`
: Server-side database will be named `<STRING>`. These two options are synonymous. For testing only. See below.

`--client-database-name <STRING>`
`--gwirdcli-database-name <STRING>`
: Client-side database will be named `<STRING>`. These two options are synonymous. For testing only. See below.

`--prefix <STRING>`
: The prefix specified by `<STRING>` is created. That is, a row is created for it in the `nas` table of the handles database. If no prefix is specified, the default prefix '`12345`' is created. This option may be used multiple times.

`--institute <STRING>`
: Database entries for institute `<STRING>` will be created. This option may be used multiple times.

`--no-standalone-handle`
: If used, a database is set up for use by an "official" handle service using a prefix or prefixes registered with CNRI. See Chapter 3 [Standalone handle service], page 9.

`--irods-server-port <STRING>`
: `setupdbs` will contact the iRODS server through port `<INTEGER>`. Default: 1247.

`--help` Prints help message and exits.

`--version`
: Prints version information and exits.

The options '`--handles-database-name`', '`--server-database-name`', '`--gwirdsif-database-name`', '`--client-database-name`', '`--gwirdcli-database-name`' should only be used for testing at the present time, because the default names, i.e, `handlesystem` or `handlesystem_standalone`, `gwirdsif` and `gwirdcli`, are currently "hardwired" in `gwirdsif` and `gwirdcli`. That is, these names are used literally to access the databases. Nor does the author see any reason to change this. However, for testing `setupdbs`, it is useful to be able to specify different names, in order not to destroy or corrupt the existing databases.

Example:

```
./setupdbs --drop --handles-database-name XXX \
   --server-database-name YYY --client-database-name ZZZ \
   --institute "YYYY:GWDG Test Institute 1" \
   --institute "ZZZZ:GWDG Test Institute 2" \
   --admin lfinsto --user jdoe --user jsmith \
   --prefix 12345 --prefix 66666 \
   --group test_group_1 --group test_group_2
⇒
Entering 'setupdbs' ('main').
Databases don't already exist.  Will create.
Handles database "XXX" doesn't already exist.
Server-side database "YYY" doesn't already exist.
Client-side database "ZZZ" doesn't already exist.
Handles database "XXX" doesn't already exist.  Creating.
Server-side database "YYY" doesn't already exist.  Creating.
Client-side database "ZZZ" doesn't already exist.  Creating.
[setupdbs]  In 'main':  'handles_database_created' == 'true'.  \
   Will create tables in 'XXX' database.
[setupdbs]  In 'main':  'create_tables_handles' succeeded, returning 0.
Created tables in 'XXX' database successfully.
[setupdbs]  In 'main':  'gwirdsif_database_created' == 'true'.  \
   Will create tables in 'YYY' database.
```

```
[setupdbs]  In 'main':  'create_tables_gwirdsif' succeeded, returning 0.
Created tables in 'YYY' database successfully.
[setupdbs]  In 'main':  'create_tables_archive' succeeded, returning 0.
Created tables for archive objects in 'YYY' database successfully.
[setupdbs]  In 'main':  'create_tables_dublin_core' succeeded, \
   returning 0.
Created tables for archive objects in 'YYY' database successfully.
[setupdbs]  In 'main':  'gwirdcli_database_created' == 'true'.  \
   Will create tables in 'ZZZ' database.
[setupdbs]  In 'main':  'create_tables_gwirdcli' succeeded, returning 0.
Created tables in 'ZZZ' database successfully.
Exiting 'setupdbs' successfully with exit status 0.
Showing tables from ZZZ database:

Users
handles
nas
```

32.5.2 Global variables

The following global variables are defined in 'gwrdifpk-1.0/src/stpclopt.web':

```
string handles_database_name = "handlesystem_standalone"
string gwirdsif_database_name = "gwirdsif"
string gwirdcli_database_name = "gwirdcli"
bool handles_database_exists = false
bool gwirdsif_database_exists = false
bool gwirdcli_database_exists = false
bool handles_database_created = false
bool gwirdsif_database_created = false
bool gwirdcli_database_created = false
bool drop = false
vector<string> prefix_vector
vector<pair<string, string> > institute_vector
vector<string> admin_vector
vector<string> user_vector
vector<string> group_vector
ofstream out_strm
ifstream license_strm
int irods_server_port = 1247
string nas_table_create_str
string handles_table_create_str
string Users_table_create_str
string handles_table_alter_str[2]
```

```
string nas_table_insert_str
string Users_table_insert_str
```

32.5.3 Functions

`int main (int argc, char *argv[])` [Main function]
Defined in 'gwrdifpk-1.0/src/setupdbs.web'.

`void finish (void)` [Exit handler]
Defined in 'gwrdifpk-1.0/src/setupdbs.web'.

`int process_command_line_options (int argc, char *argv[])` [Function]
Defined in 'gwrdifpk-1.0/src/stpclopt.web'.

`void delete_and_clear (MYSQL_RES **result_array, size_t result_array_size, vector<unsigned int *> &row_ctr_vector, vector<unsigned int *> &field_ctr_vector, vector<long int *> &affected_rows_vector, vector<string> &query_vector)` [Function]
Defined in 'gwrdifpk-1.0/src/stpclopt.web'.

`int create_databases (Scan_Parse_Parameter_Type &param)` [Function]
Creates databases.
Defined in 'gwrdifpk-1.0/src/stpcrdbs.web'.

`int create_tables_handles (Scan_Parse_Parameter_Type &param)` [Function]
Create handle database tables. The default name is '`handlesystem`', if the option '`--no-standalone-handle`' is used, otherwise it's '`handlesystem_standalone`'. See Section 32.5.1 [Invoking setupdbs], page 186, above.
Defined in 'gwrdifpk-1.0/src/stpcdhdl.web'.

`int create_tables_gwirdsif (Scan_Parse_Parameter_Type &param)` [Function]
Create server-side database tables (default name '`gwirdsif`').
Defined in 'gwrdifpk-1.0/src/stpcdsif.web'.

`int create_tables_archive (Scan_Parse_Parameter_Type &param)` [Function]
Create database tables for archive objects (i.e., iRODS objects) in the server-side database (default name '`gwirdsif`').
Defined in 'gwrdifpk-1.0/src/stparobj.web'.

`int create_tables_dublin_core (Scan_Parse_Parameter_Type &param)` [Function]
Create database tables for Dublin Core metadata in the server-side database (default name '`gwirdsif`').
Defined in 'gwrdifpk-1.0/src/stpdblcr.web'.

`int create_tables_gwirdcli (Scan_Parse_Parameter_Type &param)` [Function]

Create client-side database tables (default name '`gwirdcli`').

Defined in '`gwrdifpk-1.0/src/stpcdcli.web`'.

32.5.4 Source files

Source files (located in '`gwrdifpk-1.0/src`':

'`setupdbs.web`'

'`stpclopt.web`'

'`stparobj.web`'

'`stpcdcli.web`'

'`stpcdhdl.web`'

'`stpcdsif.web`'

'`stpcrdbs.web`'

'`stpdbcwv.web`' "Driver" file for `cweave`. It contains no C++ code. See Section 1.4 [Source code and CWEB], page 3.

'`stpdblcr.web`'

32.6 Test signals (sig_test)

`sig_test` is a simple C++ program for testing signals. The source code is located in '`gwrdifpk-1.0/src/sig_test.c++`'. See Section 6.2.6 [Signal options], page 67, Section 8.9 [Raising signals], page 92, Section 11.3 [Signal handling gwirdsif], page 99, and Section 26.11 [Signal handlers], page 155.

33 Shellscripts and Utilities

33.1 GPG keys

`output_passphrase.sh` [shellscript]

`start_gwirdsif_with_passphrase.sh` [shellscript]

33.2 X.509 certificates

`gen_x509_cert_key_pair.sh` *USERNAME COMMON_NAME* [*OPTIONAL ARGUMENTS*] [shellscript]

Optional arguments:

Email address
: Default: `nobody@nowhere.de`

Country code
: Default: DE (Germany)

Organization
: Default: GWDG

Organizational unit
: Default: gwrdifpk

Locality
: Default: Goettingen

State or province name
: Default: Niedersachsen (Lower Saxony)

User ID
: Default: Highest value retrieved from the `gwirdsif.Users` database table plus 1. If this fails, the value 1000 is used instead.

Days until expiration
: Default: 1001111 (the maximum number `certtool` will accept).

Please note: This shellscript is not "bullet-proof".

33.3 iRODS passwords (Shellscripts and Utilities)

`update_irods_passwd.sh` [shellscript]

34 Emacs-Lisp files

GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

GNU General Public License

Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source.

The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance.

However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so

available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

Variable Index

(

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Data Type Index

C

D

G

H

I

P

R

S

U

X

Function Index

(

~

A

C

D

E

F

G

H

I

L

M

O

P

R

S

U

V

W

X

Y

Z

Concept Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X