Das2 PyServer 0.3 - User's Reference

Revision 2017-07-27 Draft Work in Progress

Purpose	Who	Date
Initial Draft	C. Piker	2015-03-18
Updates to allow for hourly and yearly cache blocks	C. Piker	2017-05-19
Updates to align with current versions for das2 library and tools	C. Piker	2017-07-28

Recent Revisions

Dept. Of Physics & Astronomy
The University of Iowa
Iowa City, IA 52242

Table of Contents

Acronym	Meaning
CGI	Common Gateway Interface, A definition of how programs invoked via a Web-Server should interact with that Web-server. See RFC-3875 at http://www.ietf.org/rfc/rfc3875 .
DSDF	Data Set Descriptor File, A file defining a Das 2.2 data source for use via a Das2 server. See the "Das 2.2 Interface Reference" at http://das2.org/ .
HTTP	HyperText Transfer Protocol, The basic transport protocol for most of the world's data.
NFS	Network File System - A longstanding Linux/Unix file sharing protocol

Table 1: Acronyms used within this document

Issue	Affects
Windows Installation Instructions have not been written	Section 2.2

Table 2: Known document Issues

Document	Purpose	Location
Das2: Interface Control Document	Defines the Das2 System	http://das2.org
Das2: DasCore Library Reference	Developer Documentation for writing Java Based Das2 Client programs	http://autoplot.org

Table 3: Related Documents

1: Overview

(Some rehash of Das2 ICD, with emphasis on server side operations and caching. Give some advice to Das2 Reader implementers. Mention supporting NASA contracts and credit Larry, Ed and Jeremy with initial work)

2: Installation

Compilation and installation of the Das2 PyServer software currently requires a Linux environment. That can change, it's just not been tested. In this section a '**$**' character is used at the beginning of a line to indicate commands to run in a bash shell, and '**>**' character indicates commands to run in a Windows `%COMSPEC%` shell. It is assumed throughout these instructions that the **bash** shell is used on Linux systems and that **cmd.exe** is used on Windows.

2.1: Linux Installation

A) Dependencies

Insure that the following programs are installed on your system before beginning the installation procedure.

- GCC - Tested with version 4.4, will likely work with much older versions
- Python2.6 or higher 2.X series version
- Apache 2 - Tested with 2.2, will likely work with any 2 series Apache installation.

In addition to these general programs, the following packages are required for building and running a Das2 PyServer:

- **Linux Mint 17 and Debian 9.1**

```
$ apt-get install python-dev
$ apt-get install zlib1g-dev
$ apt-get install libfftw3-dev
$ apt-get install redis-server   (optional, required for auto-caching)
$ apt-get install python-hiredis (optional, required for auto-caching)
```

- **CentOS 7**

```
$ yum install python-devel
$ yum install zlib-devel
$ yum --enablerepo=epel install redis     (optional, required for auto-caching)
$ yum --enablerepo=epel install hiredis   (optional, required for auto-caching)
$ yum --enablerepo=epel install python-redis (optional, req. for auto-caching)
```

- **CentOS 6**

  Same package list as CentOS 7, except that python-redis has to be built manually. The python-redis version on CentOS 6 is over 4 years old and contains quite a few bugs. After installing all other packages, download and install python-redis from:

  `https://pypi.python.org/pypi/redis`

B) Setup your environment

Issue bash commands similar to the ones below:

```
$ export PREFIX=/top/level/install/dir
$ export SERVER_ROOT=$PREFIX
```

For example:

```
$ export SERVER_ROOT=/usr/local/das2srv  #Example
```

If the account under which your are going to run "make install" commands can't write to the PREFIX directory, make it manually and 'chown' as necessary. This directory will be referred to as $PREFIX in the text below.

```
$ export N_ARCH=SOME_OS_SPECIFIC_STRING
```

This is used to indicate the native architecture of your computer, there are no magic values they are just strings used to generate directory names. In NFS environments this is needed to keep different binaries separated. Though you may not be using NFS, the makefiles are constructed to take such environments into account. The value should be a lower case string without spaces. Some examples:

```
$ export N_ARCH=solaris10  #Example
$ export N_ARCH=centos6    #Example
$ export N_ARCH=debian9    #Example
$ export N_ARCH=mint17     #Example
```

The following is required during the build step to make sure different das2 servers on the same architecture occupy different build directory spaces:

```
$ export SERVER_ID=your_server_name
```

At Iowa we have the Das2 servers 'planet', and 'star' thus the values we use for the SERVER_ID variable were:

```
$ export SERVER_ID=planet  #Example
$ export SERVER_ID=star    #Example
```

Make up whatever ID is appropriate for your site, as long as it contains no spaces, any value is fine.

C) Building and Installing libdas2

The Das2 PyServer depends on a the core Das2 C library. This library provides functions for reading and writing Das2 streams, manilpulating units and other general utilities. Use the following commands to download and install libdas2.

```
$ svn co https://saturn.physics.uiowa.edu/svn/das2/core/stable/libdas2
$ cd libdas2
$ make test
$ make install
$ cd ..
```

To see the pieces you have so far:

```
$ ls $PREFIX/include $PREFIX/lib/$N_ARCH
```

D) Building and installing Das2 Stream utility programs

Server side data reduction is handled by filter programs which take a Das2 stream on standard input and produce an altered Das2 stream on standard output. Use the following commands to download and install the Das2 stream tools.

```
$ svn co https://saturn.physics.uiowa.edu/svn/das2/readers/stable/tools
$ cd tools
$ make
$ make test
$ make install
$ cd ..
```

To see the pieces installed in this step:

```
$ ls $PREFIX/bin/$N_ARCH
```

E) Building and Installing PyServer

The server itself is primarily driven by three primary programs. A python request handler CGI script, `das2_srvcgi_main`, a log handler CGI script, `das2_srvcgi_log`, and a background processing worker program, `das2_srv_arbiter`. To install these programs and their supporting libaries as well as initialize the server working area issue the following commands.

```
$ svn co https://saturn.physics.uiowa.edu/svn/das2/servers/stable/pyserver
$ cd pyserver
$ make
$ make install
```

If server-side image creation via Autoplot is desired also issue the following:

```
$ make apimg            (currently not working in Rev 9965)
$ make install_apimg    (currently not working in Rev 9965)
```

To see the pieces installed in this step:

```
$ ls $PREFIX/bin $PREFIX/lib/python*/das2server $PREFIX/etc
```

F) Link CGI scripts under Apache CGI directory

The top-level CGI programs need to be linked under some directory that is allowed to execute CGI scripts in your Apache2 configuration. To find and/or setup a CGI directory under Apache2 consult the web server configuration files for your operating system. For Debian based systems this should be under `/etc/apache2`. For RedHat based systems the configuration is under `/etc/httpd`.

It is recommended that a CGI directory named `das2` be configured on your server instead of just using the default path `cgi-bin`.

```
$ cd YOUR/APACHE/CGI/DIR
```

```
$ ln -s $PREFIX/bin/das2_srvcgi_main server
$ ln -s $PREFIX/bin/das2_srvcgi_logrdr log
```

The names 'server' and 'log' above can be any name you like. They will form part of the root URL of your server so pick names that make sense for your setup. Currently these instructions have only been tested under http:// URLs. Though there is no known reason that they will not work for https:// roots.

Since we are symlinking the top level scripts from the CGI directory instead of installing them in that location, the Apache `<Directory>` block for your CGI directory will need to allow following symlinks. This can be done by setting the directory options as follows:

```
Options +ExecCGI +FollowSymLinks
```

G) Configure the Das2 Server Program

1. First setup the peers file

```
$ cd $PREFIX/etc
$ cp das2peers.ini.example das2peers.ini
```

and edit das2peers.ini to taste. Note this is read as a UTF-8 file so go ahead and add non 7-bit ASCII characters if you like.

> *Note: Das2 was developed by English speaking programmers, and thus has not been tested under non English locales. Despite this history, it is the desire of the Das2 developers that the software should work with labels and descriptions in any language and thus UTF-8 is taken as the default encoding for all text handling. Please report any problems you encounter with text containing unicode characters at code points above 127. Bugs of this nature will be dealt with promptly.*

2. Next setup the config file

```
$ cd $PREFIX/etc
$ cp das2server.conf.example das2server.conf
```

Point `DSDF_ROOT` to the top of your DSDF file set.

Point `LOG_PATH` somewhere that Apache can write

3. If you have DSDFs with 'server' keywords embedded and you'd like to ignore these in order to test the new server, set `IGNORE_REDIRECT` to true.

4. If you want to enable server-side plot generation un-comment:

```
PNG_MAKER = autoplot_url2png.py
```

in the das2server.conf configuration file and:

```
$ edit $PREFIX/bin/$N_ARCH/autoplot_url2png.py
```

to update any paths that might not make sense on your system. Any program can be used as a server side PNG creator as long as it follows the interface defined in Appendix C.

Currently the configuration file options:

`DSID_ROOT`, `LIB_PATH` and `VIEWLOG2_RELPATH`

don't do anything, you can comment them out or just ignore them.

H) Add a Logo

Copy any file with the pattern logo.* to:

`$PREFIX/servers/$SERVER_NAME/resources`

The program will pick it up and even guess the mime-type.

2.2: Windows Installation

TBD

2.3: Testing

For your first query try:

http://hostname/your-cgi-dir/das2Sever?server=debug

The server will ask for authentication

At this point you have a server, but no data readers

3: Authentication and Security

Everyone get's confused by the definition of HTTP authentication and why the web-server doesn't just handle all authentication for Das2 server dataset access. So add a section explaining authentication and security configuration.

3.1: Setup HTTP Basic Authentication Gateway

If you want to protect some or all of your data sets via a username and password then the PyServer program will need access to this information. By default Apache does not pass authentication information to CGI scripts. To change this, open your Apache configuration file:

```
$ edit /etc/httpd/conf/httpd.conf   (for example)
```

and make the following change to your Apache cgi-bin directory.

```
<Directory "/var/www/cgi-bin">     (Your cgi directory may be different)
  # Added to pass HTTP auth tokens through to the Das2 PyServer
  RewriteRule ^ - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
  RewriteEngine on
   ...
</Directory>
```

After making the changes restart Apache

```
$ sudo /etc/init.d/httpd restart      (for example)
```

4: Caching Reduced Data

Typically Das2 Servers exist to feed data to graphical clients. In many situations the data stored on disk are more detailed that an application can display. Take for example a magnetometer than produces one measurement every quarter of a second. Typically a scientist will want to view all data for a particular day, or maybe all data for a multi-day orbit. For our hypothetical instrument, a day's measurements are 345,600 data points. Since an average computer screen is only around 2000 pixels wide sending a whole day's worth of measurements at intrinsic resolution is a waste of network bandwidth and client memory.

As part of the Das2 Client-Server protocol (see the Das 2.2 ICD), client programs may inform the server of the minimum resolution that is expected with each data request. To make better use of resources the Das2 servers use the resolution information to reduce data streams on the server before transmission to the remote client. This makes life easier on the client program, however it does nothing to conserve compute resources on the server. All relevant high resolution files must be read by reader programs for each request, and reduction calculations must be re-run for each client request. To solve this problem, the Das2 PyServer can be directed to cache reduced data streams.

4.1: Enabling a data cache

Caching is controlled via keywords in the DSDF file that defines each data source on the Das2 server. The keywords that affect the data cache are given in table 4.1 below.

DSDF Keyword	Purpose
reader	Defines the program to run to produce intrinsic resolution data.
reducer	This is the program which generates the reduced resolution datasets. If not specified a default will be used, based off the stream type. If set to `not_reducible`, then only `intrinsic` resolution caches may be created.
das2Stream	Let's the server know that the data set reader produces Das2 Streams. Different default reducers, cache writers an cache readers are invoked for different stream types.
qstream	Let's the server know that the data set reader outputs information in QStreams format. Again this affects the default reducer and cacheReader.
cacheLevel_XX	Defines the number of resolution levels in the cache and the file storage method used for each one. Each level may specify a resolution in seconds, or `intrinsic` to indicate that reader output should be cached, but not reduced.
cacheReader	This option is rarely used. It allows an alternate cache reader program to be specified.

Table 4.1: DSDF keywords involved in server side caching

4.2: Cache Layout

The following rules are used by Das2 PyServer to store and retrieve files from the disk cache. These are provided in case that you need to manually manipulate the cache, or for writing replacement cache readers.

A) Cache File Path Components

Reduced resolution files are found using the following directory path:

```
CACHE_ROOT / dataset_name / norm_params / res_dir / block_dirs / block_file
```

Table 4.2 below breaks out each component of a cache file path.

Path Component	Description	Input Source
CACHE_ROOT	The root of the pre-reduced data file tree	das2server.conf
dataset_name	The dataset name on the server. Note that dataset names are hierarchical. Each section of the name becomes a path component.	client query
param_string	This is a **normalized** parameter string, thus all spaces and other dangerous components have been transformed into alternate characters. This string does **not** include the data range query paramentes (typically a begin and end time). See part B of this section for normalization rules.	client query
res_dir	This directory corresponds to the resolution of the reduced data. • For binned data, the name is the prefix **`bin-`** concatenated with the resolution value (no units). • The server can also cache intrinsic resolution data. In this cane the resolution directory is just named '**`intrinsic`**'. This is a useful optimization when dealing with slow reader programs.	dataset DSDF file
block_dirs	These path components are based on the resolution's coverage blocks. For example if the '`daily`' coverage blocks are selected for a resolution in the DSDF file then this would be just be a single directory containing the 4-digit year. See part C of this section for the available schemes.	dataset DSDF file
block_file	The filename is produced as follows: *`coverage_resolution.suffix`* where *`coverage`* is scheme dependent, *`resolution`* is the same string found in the **resolution_dir** component above, and suffix depends on the file format, either `d2s` or `qds` for Das2 streams and QStreams respectively. See part C of this section for the available coverage blocks.	dataset DSDF file

Table 4.2: Cache File Path Components

B) Normalized Parameter Strings

todo: document the normalization algorithm

C) Available Coverage Blocks

Currently only time series data may be arranged into coverage periods. This meets must current needs but should be expanded to handle measurements such as radar returns that correlate with locations on a planetary surfaces.

Coverage Blocks	Directory Pattern	File Name Prefix	Partial Path Example
perminute	YYYY/MM/DD/HH	YYYY-MM-DDTHH-MM_	**2013/10/31/01/**2013-10-31T01-05_bin-1s.d2s
hourly	YYYY/MM/DD	YYYY-MM-DDTHH_	**2013/10/31/**2013-10-31T01_bin-1s.d2s
daily	YYYY/MM	YYYY-MM-DD_	**2013/10/**2013-10-31_bin-60s.d2s
monthly	YYYY	YYYY-MM_	**2013/**2013-10_bin-3600s.d2s
yearly	(none)	YYYY_	2013_bin-86400s.d2s

Table 4.3: Times Series Cache Blocks

Das 2.2 servers are heavily targeted towards handling time series data. It is anticipated that new coverage block storage schemes will be needed to handle future Das 2.3 client queries.

5: Server Files

5.1: Main Configuration: das2server.conf

All Das2 PyServer side programs whose name begins with 'das2_srv' use the das2server.conf file to direct some or all of their actions. The location of this file for a particular Das2 PyServer is compiled into these programs during installation. The location of the file is:

```
$PREFIX/servers/$SERVER_NAME/etc/das2server.conf
```

Here $PREFIX and $SERVER_NAME are environment variables that were specified during installation and have no default values.

The server also depends various utility programs as well to handle tasks such as converting data streams to ASCII text, up-converting old Das 1 reader output to Das 2.2 format, and reducing data streams on the server before transmission to the client. Utility programs do not consult das2server.conf, but instead must receive all necessary configuration information on the command line.

The only hard-coded path on the server is the configuration file itself. All other locations, including the dataset root directory, the logging directory, the cache location and even the program and library paths are specified in the configuration file. The server is so flexible that the configuration file can even be used to swap in custom code for parts of the main CGI routine! Flexibility like this can break things. If you are having problem's on your server first consult the logs. If that isn't helpful, try using a web-browser to issue a debug query to the server which will look similar to:

```
http://your.host.top/cgi/path/das2Server?server=debug
```

This will cause PyServer to dump it's configuration information to your browser.

SITE_IDENTITY The name of this Das2 Server. Will be provided to clients that preform a Server ID query on your site.
Default: None

DSDF_ROOT The path on your local file system to the top of the Data Set Definition file tree.
Default: $PREFIX/servers/$SERVER_NAME/datasets

CACHE_ROOT The path on your local file system to the top of the pre-reduced data set cache.
Default: $PREFIX/servers/$SERVER_NAME/datasets

RESOURCE_ROOT The path on your local file system to the top of the static file resource tree.
Default: $PREFIX/servers/$SERVER_NAME/resources

USER_PASSWD The path on your local file system to the user account password file
Default: $PREFIX/servers/$SERVER_NAME/etc/passwd

USER_GROUP The path on your local file system to the user account group file
Default: $PREFIX/servers/$SERVER_NAME/etc/group

TEST_FROM
Default: None

MODULE_PATH

BIN_PATH

LIB_PATH

LOG_PATH

VIEW_LOG_URL

VIEW_LOG_ALLOW

D2S_REDUCER

D2S_CACHE_RDR

D2S_TOASCII

QDS_REDUCER

QDS_CACHE_RDR

QDS_TOASCII

WORK_QUEUE_BROKER

WORK_QUEUE_CONN

SAMPLE_DSDF

SAMPLE_START

SAMPLE_END

INGNORE_REDIRECT

PNG_MAKER

5.2: Data Set Definition Files: *.dsdf

5.3: Server Peers: das2peers.ini

5.4: Authentication: passwd, group

6: Server Programs

6.1: das2_svrcgi_main

Purpose:	Main Das2 PyServer CGI program
Usage:	**das2_srvcgi_main** (All options are set via CGI/1.1 Environment Variables)
Scope:	Server Specific Program
Config:	das2server.conf, data set DSDF files

das2_svrcgi_main is meant to be invoked by a web-server that supports the CGI/1.1 interface standard. Most of the program's activity is directed by two inputs:

1. The CGI environment variables associated with a client request
2. The contents of the das2server.conf file

At present the POST method is not handled by the program, so standard input is not read. Furthermore das2_svrcgi_main does not write to standard error. All error messages out output as HTTP responses on standard output, as is expected of a CGI program. Configuration file details are provided in Section 5, environment variables used by this program are listed below.

Environment Variable	Summary
`HTTP_AUTHORIZATION` (optional)	Used to authenticate client programs if a data source has the keyword `readAccess` in it's DSDF file. (See Das2.2 ICD, Table 5)
`HTTP_USER_AGENT`	Used to determine if error messages should be output as Das2 Stream comments or as plain text.
`PATH_INFO`	Used to determine which resource file a client program is trying to access. (Das 2.3 protocol will use this for data source location as well)
`QUERY_STRING`	Used to determine which action the client is requesting
`REMOTE_ADDR`	Used to save log messages by remote IP address and for allowing some IP addresses to skip authentication all together
`REQUEST_METHOD`	Current version merely checks to make sure POST is not in use.
`SCRIPT_NAME`	Used to determine the URL to the Das2 PyServer itself
`SERVER_NAME`	Used to determine the URL to the Das2 PyServer itself
`SERVER_PORT`	Used to determine the URL to the Das2 PyServer itself
`SERVER_SIGNATURE`	Used to advertise the underlying web-server software on auto-generated web pages.

6.2: das2_srvcgi_logrdr

Purpose:	Log Reader CGI program
Usage:	**das2_srvcgi_logrdr** (All options are set via CGI/1.1 Environment Variables)
Scope:	Server specific program
Config:	das2server.conf

6.3: das2_srv_arbiter

Purpose:	Server background processing manager
Usage:	**das2_srv_arbiter** [-h] [--no-daemon]
Scope:	Server specific program
Config:	das2server.conf

Handles Jobs received from the task queue broker. The format

6.4: das2_srv_todo

Purpose:	Adds tasks to a Das2 server's task lisk
Usage:	**das2_srv_todo** [-h] [--no-daemon]
Scope:	Server specific program
Config:	das2server.conf

Handles Jobs received from the task queue broker. The format

6.5: das2_srv_passwd

Purpose:	Manage user accounts for Das2 Servers
Usage:	**das2_srv_passwd** [-h] `[-c] [-b] username [password]`
Scope:	Server Specific Program
Config:	das2server.conf

das2_srv_passwd updates the file indicated by the USER_PASSWD entry in a server's das2server.conf file. At present all password strings are stored using the CRYPT method. Das2 systems are assumed to operate in a somewhat friendly environment. Scientific data are usually not interesting to hackers, and user accounts on Das2 systems mostly exist to prevent premature publishing of data. For these reasons the crypt algorithim is considered sufficient to provide what little security is needed.

6.6: das2_ascii

6.7: das2_bin_avg

Purpose:	Reduces the size of Das2 streams by averaging over values in the first <x> plane in a stream.
Usage:	das2_bin_avg [-b BEGIN] BIN_SIZE
Scope:	General Program
Config:	(none)

6.8: das2_bin_avgsec

Purpose:	Reduces the size of Das2 streams by averaging over time
Usage:	das2_bin_avgsec [-b BEGIN] BIN_SECONDS
Scope:	General Program
Config:	(none)

das2_bin_avgsec is a classic Unix filter, reading Das 2 Streams on standard input and producing a time-reduced Das 2 stream on standard output. The program averages <y> and <yscan> data values over time, but does not preform rebinning across packet types. Only values with the same packet ID and the same plane name are averaged. Within `<yscan>` planes, only Z-values with the same Y coordinate are combined.

It is assumed that <x> plane values are time points. For this reducer, only the following <x> unit values are allowed:

us2000 - Microseconds since midnight, January 1st 2000

t2000 - Seconds since midnight, January 1st 2000

mj1958 - Days since midnight January 1st 1958

t1970 - Seconds since midnight, January 1st 1970

All time values, regardless of scale, epoch, or representation in the input stream are handled as 8-byte IEEE floating point numbers internally. ASCII times are converted internally to us2000 values.

The BIN_SECONDS parameter provides the number of seconds over which to average <y> and <yscan> plane values. Up to total of 99 <y> and <yscan> planes may exist in each packet type, and up to 99 packet types may exist in the input stream. This is a plane limit, not a limit on the total number of data vectors. <yscan> planes may contain an arbitrary number of vectors. The output stream has the same number of packet types and planes as the input stream, but presumably with many fewer time points.

Option	Summary
-b BEGIN	Instead of starting the 0th bin at the first time value received, specify a starting bin. This useful when creating pre-generated caches of binned data as it keeps the bin boundaries predictable.

6.9: das2_bin_peakavgsec

6.10: das2_cache_rdr

Purpose:	Pre-reduced dataset reader
Usage:	**das2_srv_cacherdr** [-h] `DATASET_NAME NORM_PARAMS RESOLUTION BEGIN END`
Scope:	Server specific program
Config:	das2server.conf, data set DSDF files

das2_cache_rdr selects pre-generated das2 stream files from a hierarchical data cache of pre-binned data. Full path to cache files is:

```
CACHE_ROOT/dataset_name/norm_params/resolution_dir/block_dirs
```

Parameter	Summary
DATASET_NAME	The name of the dataset, i.e. the relative path to the DSDF file from the root of the DSDF tree.
NORM_PARAMS	A normalized string representing the parameters set to the reader. The assumption is that when readers are called with different parameter sets the output data set changes. Each different parameter set is a different set of cache files. The string '`_noparam`' can be used to indicate that no parameters were given to the reader when the cache files were generated. See section , part for a description of reader parameter string normalization.
RESOLUTION	An integer providing the resolution requested. The largest bin size that does not exceed this value will be selected as the dataset. The string '`intrinsic`' can be used to select the best resolution available. Also a BINSZ of 0 may be given to select native resolution as well.
BEGIN	The starting value of the lookup parameter.
END	The ending value of the lookup parameter.

6.11: Readers

Appendix A: Installed File Manifest

All files are installed under a single location. The top level directory will be called `$PREFIX` in this list. Many of the file locations can be changed via the das2pyserver.conf file. In this example default locations are assumed. It is also assumed that libdas2 was installed as part of the instructions. Unix style directory separators are used for the manifest, though the actual path entries on Windows will have '\' characters.

A.1 General Files

These files may be used by any number of Das2 servers on a computer. These provide general resources which and are not specific to any particular Das2 server, reader or client program. However many of these files are specific to a particular operating system. The OS-specific directory names are represented by the variable `$N_ARCH` in this list, see Section 2.1, part B where `$N_ARCH` is defined.

```
$PREFIX/bin/$N_ARCH/das1_ascii
$PREFIX/bin/$N_ARCH/das1_bin_avg
$PREFIX/bin/$N_ARCH/das1_inctime
$PREFIX/bin/$N_ARCH/das2_ascii
$PREFIX/bin/$N_ARCH/das2_bin_avg
$PREFIX/bin/$N_ARCH/das2_bin_avgsec
$PREFIX/bin/$N_ARCH/das2_bin_peakavgsec
$PREFIX/bin/$N_ARCH/das2_from_das1
$PREFIX/include/das.h
$PREFIX/include/das2/buffer.h
$PREFIX/include/das2/das1.h
$PREFIX/include/das2/das2io.h
$PREFIX/include/das2/das2stream.h
$PREFIX/include/das2/descriptor.h
$PREFIX/include/das2/encoding.h
$PREFIX/include/das2/oob.h
$PREFIX/include/das2/packet.h
$PREFIX/include/das2/plane.h
$PREFIX/include/das2/processor.h
$PREFIX/include/das2/stream.h
$PREFIX/include/das2/units.h
$PREFIX/include/das2/util.h
$PREFIX/lib/$N_ARCH/libdas2.a
$PREFIX/lib/$N_ARCH/pythonX.X/_das2.so
$PREFIX/lib/pythonX.X/das2/__init__.py
$PREFIX/lib/pythonX.X/das2/daspkt.py
$PREFIX/lib/pythonX.X/das2/dastime.py
$PREFIX/lib/pythonX.X/das2/dastime.pyc

$PREFIX/lib/pythonX.X/das2server/__init__.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/debug.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsdfDescribe.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsdfImage.py
```

```
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsidDescribe.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/id.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/intro21.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/logo.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/resource.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsdfDataset.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsdfExList.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsdfList.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/dsidLevel.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/intro22.py
$PREFIX/lib/pythonX.X/das2server/defhandlers/peers.py
$PREFIX/lib/pythonX.X/das2server/util/__init__.py
$PREFIX/lib/pythonX.X/das2server/util/auth.py
$PREFIX/lib/pythonX.X/das2server/util/cache.py
$PREFIX/lib/pythonX.X/das2server/util/dsdf.py
$PREFIX/lib/pythonX.X/das2server/util/dsid.py
$PREFIX/lib/pythonX.X/das2server/util/io.py
$PREFIX/lib/pythonX.X/das2server/util/misc.py
$PREFIX/lib/pythonX.X/das2server/util/site.py
```

A.2 Server Specific Files

More than one Das2 PyServer may be installed on a single computer, each one is defined by a single instance of the configuration file das2server.conf. To keep installations separate, server specific files were placed together in a sub-directory named `$SERVER_NAME`. See Section 2.1, part E where `$SERVER_NAME` was defined. Programs below have embedded references to a particular listed das2server.conf file.

```
$PREFIX/servers/$SERVER_NAME/etc/das2server.conf
$PREFIX/servers/$SERVER_NAME/etc/das2peers.ini
$PREFIX/servers/$SERVER_NAME/etc/passwd
$PREFIX/servers/$SERVER_NAME/etc/group
$PREFIX/servers/$SERVER_NAME/bin/das2_srv_cacherdr
$PREFIX/servers/$SERVER_NAME/bin/das2_srvcgi_logrdr
$PREFIX/servers/$SERVER_NAME/bin/das2_srvcgi_main
$PREFIX/servers/$SERVER_NAME/bin/das2_srv_passwd
$PREFIX/servers/$SERVER_NAME/bin/das2_srv_testrdr.py
$PREFIX/servers/$SERVER_NAME/datasets/test.dsdf
$PREFIX/servers/$SERVER_NAME/resources/das2pyserver.css
$PREFIX/servers/$SERVER_NAME/resources/das2pyserver.xsl
$PREFIX/servers/$SERVER_NAME/cache/     (initially empty)
```

Appendix B: Background Processing Control Protocol

Section 6.3 introduced the program, `das2_svr_arbiter`, which handles background processing requested by other components of a Das2 server. It produces cache files, generates coverage plots, collects usage statistics, etc. By default the program is run as a daemon listening for job requests from a work queue broker. Currently the only work queue broker supported is the Redis server.[1] Thus all remaining information in this section will assume that Redis is in use.

B.1 Input Work List: das2_todo

After starting, **`das2_svr_arbiter`** opens a connection to the Redis server listed in the server configuration file, das2sever.conf, and issues a blocking right pop operation on the list variable, `das2_todo`. List values strings defining tasks to be handled and have the following format:

```
req_time|reqester|requester_ex|rmtreq|rmtreq_ex|user|job_cat|job_fields...
```

Task strings always start with the following sub-fields which are delimited by pipe '|' characters.

0: req_time - An ISO-8601 time and date string to at least seconds resolution, example: `2015-03-25T10:37:43.334`

1: requester - An identifier of the program making the request. For example the URL to the das2server associated with the task may be used: `http://planet.physics.uiowa.edu/das/das2Server`

2: requester_ex - Extra information about the server, may be empty.

3: rmtreq - Some programs, such as web servers handle request on behalf of remote clients. The IP of the client requesting the operation. Note this may the IP of the server itself if a work item was entered via the **`das2_srv_todo`** program.

4: rmtreq_ex - Extra information about a client, such as it's GeoIP location, may be empty.

5: user - The user name, if any associated with the work request, may be empty

6: job_cat - The job category, may not be empty. Sections B.1.1 through B.1.3 below list the current job categories and the job-specific fields

7-N: job_fields - Each category of job has it's own list of fields following the job_cat field. See Tables B.1 through B.3 for a description of each job category and the required fields.

Empty fields are represented by two adjacent pipes with no intervening characters. To make the protocol expandable, each job category has it's own field set. These are added at index 7 and continue as far as is dictated by the category.

B.1.1 Build Cache Task

This task requests a cache build for a specified range of a dataset at one, or all of the resolutions listed in the dataset's DSDF file. This task type is specific to the Das 2.2 system interface. Upcoming Das 2.3 cache tasks will presumably require a different set of fields, or different field values.

Job Category Value: **`TASK_CACHE`**

1 See http://redis.io for more information on Redis, and advanced key-value cache and store server.

Overall Field Index	Job Field Name	Job Field Description
7	dataset	The name of the dataset that needs to be cached. This is always the relative DSDF filename without the extension. Examples: `juno/wav/survey`, `voyager/1/pws/SpecAnalyzer-4s-Efield`
8	begin_index	The beginning index to cache, for Das 2.2 this is an ISO-8601 start time.
9	end_index	The ending index to cache, for Das 2.2 servers this is an ISO-8601 end time.
10	cache_level	A requested cache level index. This maybe the keyword **`ALL`**, to indicate that all cache levels specified in the dataset's DSDF file are to be produced, or this may be one of the given cache index values. See the **`cacheLevels_XX`** DSDF value in Section 3.2 of the Das 2.2.1 ICD. Leading zeros are ignored.

Table B.1: ***TASK_CACHE*** *- Task Specific Fields*

The following string provides an example of a cache_2.2 task request:

```
2015-03-25T10:37:43.334|jupiter||128.255.33.104|US,Iowa,Iowa City||TASK_CACHE|
juno/fgm/MagComponentsSCSE||2014-01-01|2014-01-02|ALL
```

B.1.2 Record Access Task

Job Category Value: **`TASK_USAGE`**

Access event tasks are meant to record which datasets are being retrieved. The Job specific fields are given in Table B.2 below.

Overall Field Index	Job Field Name	Job Field Description
7	dataset	The name of the dataset that needs to be cached. This is always the relative DSDF filename without the extension. Examples: `juno/wav/survey`, `voyager/1/pws/SpecAnalyzer-4s-Efield`
8	begin_index	The beginning index to cache, for Das 2.2 this is an ISO-8601 start time.
9	end_index	The ending index to cache, for Das 2.2 servers this is an ISO-8601 end time.
10	params *(optional)*	A non-indexing parameter string. Some readers take parameters that are not part of the start and stop period of the data cache.
11	resolution	The floating point resolution datum requested from the client, or empty if no specific resolution was specified.
12	interval	Some datasets are produced at required intervals only. If such a dataset was accessed this field records the interval, otherwise it is blank.

Table B.2: ***TASK_USAGE*** *- Task Specific Fields*

The following string provides an example of an access log request for the Voyager spectrum analyzer distogram reader:

```
2015-03-25T10:37:43.334|planet||128.255.33.104|US,Iowa||TASK_USAGE|
voyager/1/pws/SpecAnalyzer-Distogram|2011-07-24|2011-08-31|10 10|340.0 s|
```

B.1.3 Update Coverage Plot

Job Category Value: **`TASK_COVERAGE`**

PyServer can provide dataset coverage plots, this work item request an update of those inventory plots.

Overall Field Index	Job Field Name	Job Field Description
7	dataset	The name of the dataset whose coverage needs to be updated, or the keyword **ALL**, to indicate that coverage for all datasets should be updated.
8	begin_index	The beginning time for finding data coverage
9	end_index	The ending time for finding data coverage.

Table B.3: ***TASK_COVERAGE*** *- Job Specific Fields*

The following string provides an example of a massive coverage plot update.

```
2015-03-25T10:37:43.334|planet||128.255.33.71||cwp|TASK_COVERAGE|||1977-01-01|2
016-01-01
```

B.2 In-Process List: das2_working_*proc-id*

Each *instance* of das2_srv_arbiter uses a single list to record jobs in process. The current version of the arbiter is single threaded, so at most one item will appear in a `das2_working_NNNNN` list, it's format is:

```
req_time|server|server_ex|client|client_ex|user|job_cat|job_specific...|begin|
status|progress
```

The working list just adds two fields to the end of which-ever job type is in progress. These are:

-3: begin - An ISO-8601 time indicating when the job was started.

-2: status - Current job status message. *(optional)*

-1: progress - A fraction giving how much of the job is completed, where 1.0 is the entire job. The fraction may be specified as a fraction using the '/' sign to separate two numbers.

Here is an example of a coverage plot task item value.

```
2015-03-25T10:37:43.334|planet||128.255.33.71||cwp|TASK_COVERAGE|||1977-01-01|2
016-01-01|2015-03-25T10:37:44.512|Updating cassini/mag/4-vec|32/114
```

B.3 Result List: das2_finished

Finished items end up in a single list for the entire site. This list persists until cleared by a logger. These items are identical to In-Process items with the exception that the progress value is replaced with an end time.

```
req_time|server|server_ex|client|client_ex|user|job_cat|job_specific...|begin|st
atus|end|return_code
```

-4: begin - An ISO-8601 time indicating when the job was started.

-3: status - End status message. *(optional)*

-2: end - An ISO-8601 time indicating when the job finished.

-1: return_code - The finial return code from the handling the work item. To continue the tradition set long ago, 0 indicates success non-zero is a failure code.

```
2015-03-25T10:37:43.334|planet||128.255.33.71||cwp|TASK_COVERAGE|||1977-01-01|2
```

```
016-01-01|2015-03-25T10:37:44.512|Generated coverage map for all 114
datasets.|2015-03-29T01:14:55.512|0
```

Appendix C: Server-Side Plot Creation Interface

(Initial draft, not complete)

Any program that can take as it's inputs the following command line parameters:

```
server=SERVER_NAME
dataset=DSDF_RELATIVE_PATH
start_time=BEGIN
stop_time=END
image=OUTPUT_FILE_NAME
param=EXTRA_PARAMS
```

and writes a PNG file to the given location can be used to buy the das2_srvcgi_main script to generate server-side plots which are then transmitted to the client instead of a numeric dataset. As an example, the command line parameters to request an EMFISIS High Frequency Receiver Dynamic Spectrum with noise spikes removed might look like:

```
your_program dataset=rbsp/RBSP-B/HFR_spectra.dsdf  \
     start_time=2013-03-01 \
     end_time=2013-03-02 \
     image=/usr/local/das2srv/tmp/HFR_spectra_34578.png \
     params='KEEP-SPIKES Bu' \
     server=http://emfisis.physics.uiowa.edu/das/das2Server
```

An example wrapper script using the Autoplot program to create plot PNGs is provided in the source code for the server and is build when issuing the commands: "`make aping`" and "`make install_apimg`".