

UNIVERSITY OF HAWAII AT MĀNOA

Institute for Astronomy

Pan-STARRS Project Management System

Data Store

Coop. Agreement No. : FA9451-06-2-0338
Prepared For : Pan-STARRS Software
Prepared By : Josh Hoblitt, Sidik Isani, Ed Pier, Craig Rae, Erik Small
Document No. : PSDC-940-010-03
Document Date : May 7, 2008
Revision : 03

DISTRIBUTION STATEMENT

Approved for Public Release – Distribution is Unlimited

Submitted By:

[Insert Signature Block of Authorized Developer Representative]

Date

Approved By:

[Insert Signature Block of Customer Developer Representative]

Date

The information here is available as HTML. The URL is: <https://svn.ifa.hawaii.edu/gpc/archive/psdc/gpc-datastore/>

Revision History

Revision Number	Release Date	Description
00	2006.4.3	First release. Most camera-specific stuff is gone.
01	2007.9.25	Clarified index.txt HTTP paths and updated hardware prices.
02	2007.10.8	Clarified date string format.
03	2008.5.5	Added more details on HTTP Status Code usage.

Abstract

The “Data Store” is a separate hardware and software project developed by the Giga Pixel Camera group. Its function is to receive data files from one subsystem (such as Camera or skyprobe), serve them to IPP and other clients. It must also continually remove old files to maintain space for incoming files. IPP, OTIS, and others also exchange metadata by placing it in smaller FITS table data files.

This is the functional specification and design of the Data Store system. Camera-OTIS, Camera-IPP, OTIS-IPP ICDs, and others may refer to this document as it contains the details of how to access the Data Store.

Contents

1	Functionality	1
1.1	Filesystem	1
1.2	Outbound HTTP service	2
1.3	Housekeeping	2
2	Implementation	2
2.1	Filesystem	3
2.2	HTTP Service	4
2.3	Housekeeping	4
2.4	Scaling it to a GPC	5
3	Accessing the Data Store	5
3.1	Filesystem	5
3.2	HTTP Service	6
3.2.1	List Formats	7
3.2.2	File Set List Format	7
3.2.3	File List Format	8
3.2.4	Error Handling	9
3.3	Housekeeping	11
4	System Status and Maintenance	11
4.1	Hardware Status	11
4.2	Disk Maintenance and Replacement	12
5	Cost	13

List of Figures

1	Data Store Block Diagram	1
2	Data Store Design	5
3	Server Status Web Page	12

1 Functionality

The functionality of the Data Store system is driven by the data exchange needs of OTIS, Camera, and IPP.

- Camera and Skyprobe need a filesystem for storing FITS images
- OTIS and IPP need to exchange metadata dumps (also in FITS format, as FITS tables)
- IPP and OTIS need *read-only* access to each images and tables.
- The volume of data necessitates automated housekeeping.

Data Store serves as a cache between a “local” system that produces data files, and “remote” system that needs access to the files. If there are network problems between local and remote, the local end will be able to keep building up data for the remote end that can be transferred whenever the network is restored. For this reason, Data Store must exist close to the system producing data, or the “local” end of the connection.

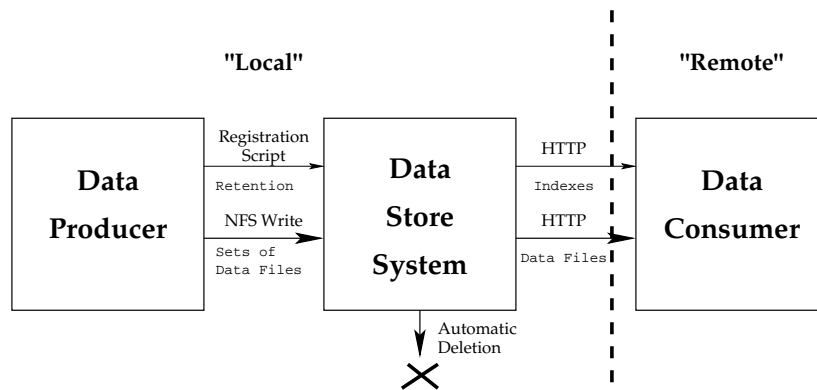


Figure 1: Data Store Block Diagram

Another important feature of the Data Store is that it retains data for a given amount of time (given when data is registered) and deletes it as needed when the retention period has expired. The retention period is a guaranteed minimum. Files may be kept longer if possible, but the remote client can only rely on being able to request files for certain during this retention period. The period (or time to live) is counted from each file’s modification time.

1.1 Filesystem

Clients with access to the Data Store’s filesystem (that is, local clients) must use NFS, and can expect to have available all the functionality of NFS version 3. No other network filesystems or protocols will be supported. NFS 3 supports changing ownership, permissions, and attributes, creating files, appending, removing, and locking. Once registered, files will appear on HTTP directory indexes for remote clients. They must not be removed or modified by NFS clients. There is no enforcement of this. Files that are never registered remain invisible to remote clients. In this way, local clients are free to use the Data Store as “normal” disk space. Remote clients, on the other hand, are never permitted to access the Data Store through NFS. (They use only the HTTP service described in the next section.)

Special consideration must be given to NFS access in the case of the hybrid Data Store nodes used at the summit which cache Giga Pixel FITS data. These hybrid nodes also serve as Pixel Servers managing the CCD controllers. This is a very CPU intensive task, during which extraneous heavy access to NFS by local clients could be detrimental to observing efficiency. For this reason, the OTIS system will usually function as a “remote” client to these Data Store nodes since local access, while permitted, will not be throttled in any way.

OTIS has voluntarily decided not to use NFS access to the data store, and so as of this writing we have no intention of implementing any NFS access.

1.2 Outbound HTTP service

The HTTP service is much more restrictive than the NFS interface to Data Store. Most notably, it is *read-only*, and there is no effect on file retention whether a client ever looks at the file or not. In the case of the hybrid Data Store/Pixel Server nodes, bandwidth *is* restricted as necessary, to guarantee optimal Camera performance. In addition, all Data Store HTTP servers have these features:

- Access restriction by IP network address.
- Provider of data makes up a unique FilesetID and FileIDs (*not* a “ticket” system.)
- Includes “Index” service
 - Can list all FilesetIDs that are available (optionally: all those since a given FilesetID.)
 - Can list all FileIDs, given an FilesetID
 - Can return the file itself, given a FileID.

Note that the index is not a directory listing. The service lists files that have been registered, not everything that might be on the disk. In fact, it will also list files which once existed but are no longer available.

1.3 Housekeeping

Whether registered or not, all files on the Data Store’s disk are subject to removal once their retention period is over. A built-in processes that is part of the Data Store continually deletes these files as needed.

Only a local client with NFS access can influence the behavior of the housekeeping daemon. It may do so by:

- Changing the group ownership of file(s)
- Updating the time of a file to renew it (“touch”)
- Freeing unregistered files to delay cleanup of expired registered files.

2 Implementation

Each subproject requiring data exchange through the Data Store mechanism will have hardware assigned for this purpose that must be installed near the source of the data.

As a cost saving measure, and as a matter of efficiency, the summit Data Store disk space will be combined with the 1U "Pixel Server" rackmount processing systems necessary to operate a Giga Pixel camera. This is accomplished by using rackmount servers with a high density of hot swappable harddrive bays for the pixel servers and filling them with large, cheap PC disks (4x750 GB SATA per 1U is our implementation at the moment.)

In order to function as a Data Store, the Pixel Servers will require the following:

- A second Ethernet interface, to avoid contention with STARGRASP communications.
- Network routing and access through any firewalls to IPP and other Data Store clients.
- Large filesystem support with local and NFS access (usually a Pixel Server will write data to the part of the Data Store that it comprises, but this is not always the case.)
- A process for HTTP service.
- A process to monitor disk usage and perform housekeeping tasks.

The last two processes shall be required to monitor their own impact on Pixel Server functions during both guiding and readout phases. They must scale back disk i/o and network i/o as necessary to avoid interfering. This is an additional burden resulting from combining the Data Store and Pixel Server functions, but is justified by the significant savings in hardware cost.

Separate nodes not used as pixel servers will exist for dedicated IPP-OTIS OTIS-IPP and skyprobe storage. In practice, the Data Store will effectively be partitioned with appropriate sizes for camera data, skyprobe data, and dbdump data. Data Store nodes for data that does not originate at the summit will be located off-summit, close to their local data producers. This is also true for the Data Store node which carries FITS table communications from IPP back to OTIS. Since IPP is the source of the data, that Data Store will be located in the same local network as the IPP cluster.

2.1 Filesystem

Without increasing the Pixel Server size from its current 1U target, we are limited to 4 hot-swap disks per chassis. A RAID-0 configuration would result in 4xSIZE where SIZE will be 750 GB for TC3/GPC1. But RAID-0 does not provide fault tolerance (see table below). RAID-5 has been tested on 3ware SATA controllers in both hardware and software RAID-5 modes. Time to rebuild one of these RAID-5 units after a disk failure has been benchmarked at 2 hours, which is quite acceptable. Each unit configured as RAID-5 will have 1.2 TB.

XFS currently appears to be the superior filesystem choice for efficient sustained write performance. Write performance is a critical benchmark in order to be able to keep up with Pixel Server data flow during a readout so that the server can be ready for guiding operations in the next exposure cycle as quickly as possible. Again, implementation choices for the hybrid Data Store-Pixel Server machine will be driven by Pixel Server design requirements - specifically, that of being able to receive 128 Mbps per second per OTA, and a minimum 4 OTAs. Being able to handle 8 OTAs at that rate on a single Pixel Server could have significant operational benefits, such as not having to degrade performance if one Pixel Server is operating with a non-functional RAID due to a disk failure.

Hardware has been received and tests show that Software RAID performs very well. RAID-0 is the best, as expected, but is not a good choice because of required fault tolerance. The best option is Software RAID-5, so that is the planned implementation for PS1.

RAID Type	Capacity (750GB disks)	Throughput	CPU Usage	Fault tolerance
Software RAID-0	3000 GB	200 MBytes/sec = 1600 Mbps	10% CPU	No
Software RAID-1-0	1500 GB	110 MBytes/sec = 880 Mbps	15% CPU	A-
Software RAID-5	2250 GB	85(*) MBytes/sec = 920 Mbps	20% CPU	B
3w8500 RAID-5	2250 GB	15 MBytes/sec = 120 Mbps	2% CPU	B+

(*)Note: Above numbers are with 4 Seagate drives.

2.2 HTTP Service

Security will be achieved by allowing only specific IP addresses through an IP firewall at the operating system level (with an iptables firewall.)

Web server will be any standard HTTP server (apache2?) The root page must include disk usage statistics including current free space, and the proportion of files that can and cannot be deleted by housekeeping in order to maintain adequate free space for new file sets.

Index service may be implemented through a module in the server or an external CGI. Details of the interface are found in section 3.

The interim (conductor) data store is running at Pan-STARRS 1 and can be accessed from Manoa or the summit at this URL: <http://conductor/ds/>. Contact camera group software support to have your machine's IP address added to the firewall if you cannot see this page.

2.3 Housekeeping

We rely on Operating System and filesystem features to implement most of the housekeeping that will be necessary. Specifically:

- A file's time stamp will tell how old it is.
- A file's unix "group" will signify the minimum amount of time it is to be kept (and this can be modified later, through the OTIS-Camera command interface.)
- A file's unix "owner" will signify which camera it came from ("tc3", "gpc1").
- The filename will be its unique identifier, and should include who created the file ("o*" for otis, "c*" for camera engineering interface ...).

Using filesystem quotas and "df" statistics, the housekeeping process will be able to monitor the vital signs which it needs for determining when to delete files.

2.4 Scaling it to a GPC

Each Pixel Server equipped with disks runs its own Data Store services. They are independent. A separate, externally visible host runs a higher level HTTP directory service which:

- Reports worst case statistics of all data stores put together
- Tracks File set IDs
- Redirects clients to individual data stores for FITS retrieval

The following figure illustrates the use of a Data Store master which manages the registration for all Data Store RAID units, and is always the first point of contact for remote clients.

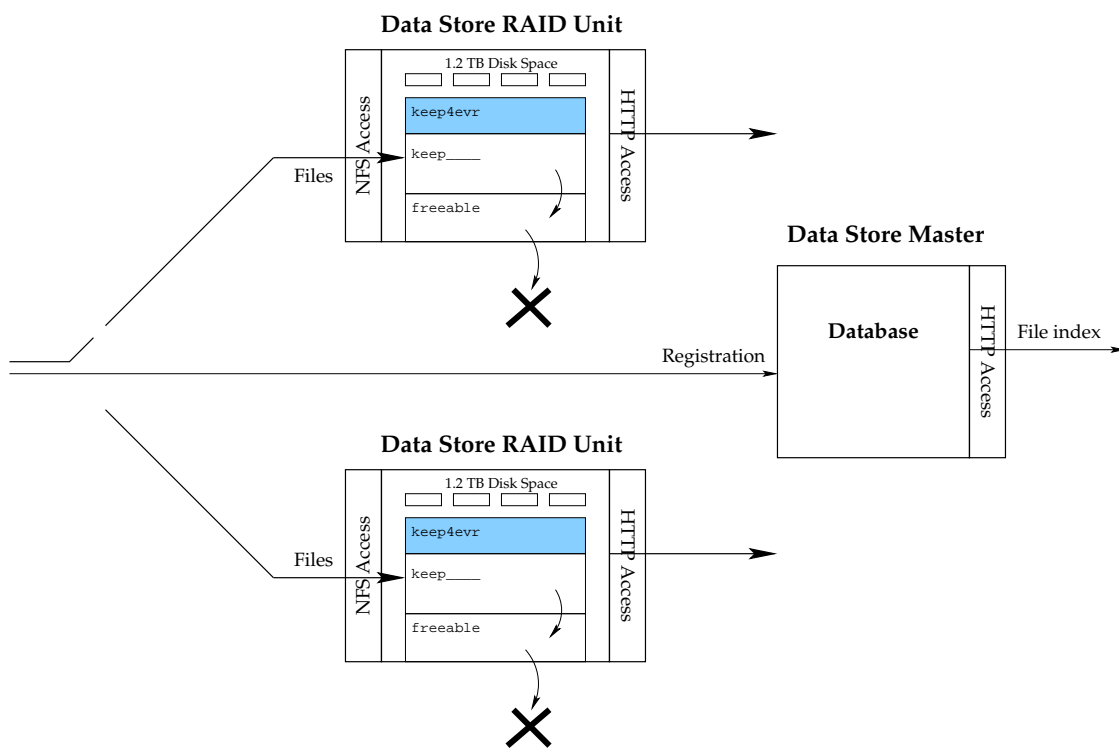


Figure 2: Data Store Design

3 Accessing the Data Store

3.1 Filesystem

Interfaces to the filesystem are provided by the network and operating system. These interfaces include unix stream IO, file descriptor IO, and memory mapping, as supported by XFS and NFS.

TBD: A script or program must be provided to allow registration of a file that has been placed on the filesystem. Without that, the HTTP server should not list a file.

3.2 HTTP Service

A normal HTTP client can be used to locate and retrieve files from the HTTP service of the Data Store.

Begin at a starting data store page of the HTTP server. We'll typically use '/ds/' instead of the root page '/' so the HTTP server can be used for things other than data store. The /ds/ page will report, in human-readable HTML form:

- Data Stores on-line and off-line.
- Worst case disk space available to a pixel server.
- Percentage of space used by each "group" (including the proportion of files marked "permanent".)
- A link to a text list of product IDs (i.e., list of cameras and databases available.)

The link to the text list of product IDs will be 'index.txt'. Robots (automated clients that expect the List Formats 3.2 described subsequently) should not attempt to parse the HTML as it may change or contain extraneous links intended for human convenience. The starting URL for an automated client should be of the form:

```
http://datastore.host.name/ds/index.txt
```

Alternatively, a bootstrap datastore URL per camera/productID will be maintained in the status server. The starting point for a particular camera, such as 'gpc1', will be contained a corresponding statserv entry:

```
ss:/datastore/all/gpc1 = "http://conductor.ifa.hawaii.edu./ds/gpc1/index.txt"
```

Adding any one of the productID's to the URL (after the final '/') returns a text list of file set IDs (one per shutter activation). For clients expecting the List Formats 3.2, 'index.txt' is appended again:

```
http://datastore.host.name/ds/productID/index.txt
```

(This ends up being the same starting URL given in the status server for the productID.)

The request for the index.txt list of Exposure IDs takes an argument which should be the most recent exposure ID you already know about. This is appended to the URL as the "query string", separated by a (real, not encoded) '?'. The list will then include only those IDs created after the one given:

```
http://datastore.host.name/ds/productID/index.txt?o1234g0356o
```

Still using the same HTTP server, a client can request a list of files associated with any fileset ID. This will result in another, similar text table. For a Giga Pixel Camera fileset, there should be 60 rows in this table, plus possibly extra rows for calcite and shack-hartmann products in the future.

A file list table shall not be returned until each of the individual files actually exists. The entry in the list of filesetID's, however, will be returned by the time the shutter is closed and files are in the process of being written. Requesting the file list for a recently published filesetID may result in the HTTP connection blocking for several seconds, while the HTTP server waits for the fileset to be completed. As with the other two lists, robot clients will want to append 'index.txt' to receive the text version:

`http://datastore.host.name/ds/productID/o1234g0357o/index.txt`

The final level of access consists of adding the fileID from the table above to the URL (after the final '/'). Example:

`http://datastore.host.name/ds/productID/o1234g0357o/o1234g0357o11.fits`

This causes the file to be returned, possibly via a redirect to another HTTP server. The redirect may be used to balance the load on several machines and/or to cause a file to be loaded directly from the host or pixel server on which it is actually stored. Such a redirect is handled transparently by any HTTP client and does not affect the logical interface to the data store.

If the previous level (file list) is bypassed and a direct request is made for a file that is not yet closed, then the result is undefined and unsupported behavior. Only the file list level request may implement the wait for file(s) that have been registers but are still being written to disk.

Note that URLs with 'index.txt' as the target file return the non-HTML version of the list page. One ending in '/' will return the HTML-formatted page with clickable links.

3.2.1 List Formats

Both File Set and File lists shall be in a plain ASCII (not HTML) table with pipe-separated (|) fields and newline (\n) terminated rows. The pipe character (|) and hash character (#) are reserved and cannot be used within any of the fields. Blank lines are also ignored. There is no character escape mechanism. Leading or trailing whitespace around a field is ignored. Embedded whitespace is preserved. All rows where the first non-whitespace character is a hash are consider comment lines and are to be ignored.

Unless otherwise noted all list fields are restricted to the format [a-z0-9-_.:] and limited to a maximum of 64 characters in length.

3.2.2 File Set List Format

The basic number of fields in this list is variable depending on the product but will always contains the following mandatory fields: `filesetID`, `datetime`, and `type`. The mandatory fields "may" be followed by optional, product specific, fields.

Mandatory field descriptions:

- `filesetID` - unique exposure (or fileset) ID assigned by data taker. 64 char limit
- `datetime` - time that the ID was registered with data store in the ISO8601 format written here using `strftime(3)` conversion specifiers: "%Y-%m-%dT%H:%M:%SZ". Note the literal 'Z' means that all times must be in UTC.
- `type` - a type for the dataset.

For type of OBJECT, DOMEFLAT, SKYFLAT, BIAS, and DARK, these additional columns are defined:

- `telescope_pointing` - in the format RA Dec angle epoch. Where RA & Dec are in the format HH:MM:SS, angle is in degrees, and epoch is always J2000.
- `integration_time`
- `filter`
- `airmass`
- `comment` - an optional field describing the program, object and/or observer.

For example:

```
# filesetID|time registered      |type |telescope pointing          |etime|f|airmass|comment
otis0123456|2006-01-01T00:03:04Z|OBJECT|11:00:10.33  68:26:59.6 2000|30.0|r|1.23|
otis0123456|2006-01-01T00:03:04Z|OBJECT|11:00:10.33  68:26:59.6 2000|30.0|r|1.23|
otis0123456|2006-01-01T00:03:04Z|OBJECT|11:00:10.33  68:26:59.6 2000|30.0|r|1.23|
otis0123456|2006-01-01T00:03:04Z|OBJECT|11:00:10.33  68:26:59.6 2000|30.0|r|1.23|
```

3.2.3 File List Format

Whether the data is a camera image or database dump, the first columns are always:

- `fileID` - a unique file ID assigned based on the fileset ID (which, for gpc data, is assigned by OTIS).
- `bytes` - integer value
- `md5sum` - hex encoded
- `type`
- zero or more type-specific columns.

For camera data, the type is “chip” and the additional columns are:

- an identifier of the chip (the OTA) unique within this camera.
- TBR: Sky background
- TBR: Other quick stats available at readout time?

For example:

```
# fileID      |bytes  |md5sum                                     |type|chipname|
otis0123456.01|83002312|fe6a2b6564c0d4cfb3bbf1db813824ba|chip|ota01|
otis0123456.02|83002312|a2b4a7d7b94dc6076c5f3f67239a48c6|chip|ota02|
otis0123456.03|83002312|a39b1510484c7833e27454b181f13981|chip|ota03|
otis0123456.04|83002312|7bb35e1e30f3f833c0416aea75f90304|chip|ota04|
otis0123456.05|83002312|fe6a2b6564c0d4cfb3bbf1db813824ba|chip|ota05|
otis0123456.06|83002312|a2b4a7d7b94dc6076c5f3f67239a48c6|chip|ota06|
otis0123456.10|83002312|7bb35e1e30f3f833c0416aea75f90304|chip|ota10|
otis0123456.11|83002312|fe6a2b6564c0d4cfb3bbf1db813824ba|chip|ota11|
```

```
otis0123456.12|83002312|a2b4a7d7b94dc6076c5f3f67239a48c6|chip|ota12|
otis0123456.13|83002312|a39b1510484c7833e27454b181f13981|chip|ota13|
...
otis0123456.74|83002312|7bb35e1e30f3f833c0416aea75f90304|chip|ota74|
otis0123456.75|83002312|fe6a2b6564c0d4cfb3bbf1db813824ba|chip|ota75|
otis0123456.76|83002312|a2b4a7d7b94dc6076c5f3f67239a48c6|chip|ota76|
```

3.2.4 Error Handling

Upon error the Data Store HTTP service will return an HTTP status code. A successful request will always return either a 2xx (OK) or 3xx (redirect) status code. Error will always be either 4xx or 5xx code. If a 4xx code is received the client is expected to treat this as a fatal error and not re-attempt the request. If a 5xx code is received the client is allowed to retry the request after 60 seconds.

Below is a partial list of status codes the client is expected to understand:

```
200 OK
302 MOVED
404 NOT FOUND
410 GONE
503 Service Unavailable
```

3.2.4.1 200 OK

This is the normal return code for a successful request. The document follows. If content length or checksums do not match, the client application should remove the file and flag this as a problem that requires attention. Optionally, if a TCP connection was broken during the download, the client could retry after a few minutes.

3.2.4.2 302 MOVED

Currently, this status is only returned for requests for a directory without a trailing slash. In the future, it will be used for FITS files as well, as a mechanism to load-balance downloads across a group of data store/firewall machines. Clients should follow the 302 status as specified in the HTTP standard (by fetching the new URL in the Location: HTTP header.)

3.2.4.3 404 NOT FOUND

Any invalid request for a location other than a removed (old) FITS file produces this standard HTTP response. The client should not retry. This return code generally signifies a programming or human error on the client or server side.

3.2.4.4 410 GONE

This is the normal return status when a request is made for a fileset which has already been purged from the data store. The client should not retry.

Here is an example of a 410 response for an exposure set index.txt file:

```
HTTP/1.0 410 Gone
Date: Tue, 06 May 2008 00:17:55 GMT
Server: RPM/3.24
Content-type: text/plain
```

The file /c4357g0346d/index.txt is either taking more than 60 seconds to appear or has been removed from the data store.

Note that the 410 response is only implemented for the text format index, **index.txt**. Automated clients may request the index.txt of an exposure set that is currently in the process of saving, and will be blocked for up to 60 seconds while the files are created on disk. This is not implemented for the **index.html** human-readable version of the data store.

More commonly in the case of an old removed exposure set, the index.txt will still be retained but the FITS files to which it refers will have been deleted. In this case, the failure will occur when requesting one of the FITS files:

```
HTTP/1.0 410 Gone
Date: Tue, 06 May 2008 00:38:36 GMT
Server: RPM/3.24
Content-type: text/plain
```

The file /c4357g0346d/c4357g0346d02.fits has been removed from data store.

In rare circumstances, “410 Gone” could also be a response to failed or aborted exposure request which never completed. In this case, the fileset entry exists in index.txt, but the files were never created due to a software error. No distinction is made, as this condition can be considered equivalent to an exposure request with a 0 retention period which was immediately removed. It may be possible to gain more information about the “410” failure by looking at the document (the content sent after the status code). For example:

```
HTTP/1.0 410 Gone
Date: Tue, 06 May 2008 01:05:47 GMT
Server: RPM/3.24
Content-type: text/plain
```

The file /o4413g0060o/o4413g0060o25.fits is corrupt or contains no data. Most likely, the exposure was aborted and not correctly saved.

3.2.4.5 503 TRY AGAIN

The requested location is temporarily unavailable. The client should retry an a reasonable period (every hour, for example). This error can occur when the requested resource is on a disk that is undergoing maintenance.

Here is an example 503 response:

```
HTTP/1.0 503 Try Again
Date: Tue, 06 May 2008 00:17:55 GMT
Server: RPM/3.24
Content-type: text/plain
```

Host t21 could not be reached for file /c4357g0346d/c4357g0346d05.fits (FILESIZE=). Please try again in about an hour. Connection closed by foreign host.

3.2.4.6 Other errors

When the primary data store HTTP server is undergoing maintenance, it is possible to see refused TCP connections. Other errors such as no route to host when the host is unreachable because it is down, or a switch or router in your path to the server is down should all be treated the same as “503 TRY AGAIN”.

3.3 Housekeeping

At any time, the OTIS system (or anyone using the Conductor command line socket) may update the time-to-live status of a file, or a group of all files within a FilesetID. This simply results in a change of “group” status which the housekeeping daemon notices and makes use of next time it needs to free up space.

Because the system works this way, it is possible that a file which should have been deleted already may still be available if the space wasn’t required for anything else. Files will never be deleted before their time to live has expired. As a result, a condition in which there is no space for a new file set is possible, and simply results in a “disk full” error. OTIS should choose between the policy options carefully, and if it still ever occurs, must re-assign current TTLs, wait a few minutes, and adjust policy use to avoid future occurrences.

4 System Status and Maintenance

The Data Store attempts to leverage operating system features for status and maintaining data integrity. The unix **df** command alone will not be very useful, except to confirm that the housekeeping task is properly maintaining the free space at a few percent of the total volume size. More information will be available by enabling filesystem **quotas**. A **quotacheck** per group provides an efficient, nearly instantaneous way to access the running totals of un-freeable data (files in the `keepperm` group), and data in and out of the retention period. The same quota system can also provide running totals of disk space per userid, i.e., separated by the original source of the data.

4.1 Hardware Status

The current status (updated every few minutes) of every Data Store node will be maintained on a Web page. Access to the Web page will be restricted by IP address. With the present hardware platform intended for Data Store, the status we can obtain includes:

- Up-time
- Ethernet link status (primary and secondary)
- Processor load
- Disk volume size and percent used
- Number of disks and state of the raid (OK/Rebuilding/Degraded)
- Temperature of motherboard, CPU, and each disk.

- Fan speeds
- Chassis intrusion

Info from DNS	Statistics from "nmap"				Statistics mostly from "/proc"									Statistics mostly from i2c									
	Up	High TCP	MAC Manuf.	Operating System	Boothost	Eth0	Eth1	CPUs	Mem	Swap	Data	Disk	Load	D1	D2	D3	D4	C1	C2	C3	F1	F2	F3
sn0[10.213.1.200] GPC / *darkroom / supernova pixserv 1U (i)	34 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 95%	4-disk ok	0.00	34	34	35	31	36	36	38	9	9	9
sn1[10.213.1.201] GPC / *darkroom / supernova pixserv 1U (i)	34 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 36%	4-disk ok	0.00	36	33	32	33	38	37	37	9	9	9
sn2[10.213.1.202] GPC / *darkroom / supernova pixserv 1U (i)	34 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	34	33	32	33	39	36	36	9	9	9
sn3[10.213.1.203] GPC / *darkroom / supernova pixserv 1U (i)	34 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	36	31	32	31	36	39	36	8	8	8
sn4[10.213.1.204] GPC / darkroom / supernova pixserv 1U (i)	36 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	35	32	30	32	39	40	37	9	9	9
sn5[10.213.1.205] GPC / darkroom / supernova pixserv 1U (i)	36 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	31	31	31	35	37	35	36	9	9	9
sn6[10.213.1.206] GPC / darkroom / supernova pixserv 1U (i)	36 days	2049	Supermicro	Linux 2.4.32-lm1	lohi	1000FD	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	31	31	32	35	39	36	36	9	9	9
sn7[10.213.1.207] GPC / darkroom / supernova pixserv 1U (i)	Down	2049	Supermicro	Linux 2.4.32-lm1	lohi	0	1000FD	3000M x 4	2017M	1953M	1117G 1%	4-disk ok	0.00	37	34	33	34	37	39	38	9	9	9

Figure 3: Server Status Web Page

A Web page will be constructed from a combination of statistics published by the Data Store node itself, and obtained by a watchdog script. The watchdog script will e-mail any significant changes to a mailing list (including if a node stops responding). Anyone responsible for at least one Data Store node must be on this list.

Hardware status for the benefit of automated software systems will be stored in a location still TBD in the shared namespace. Any system which has the ability to choose between multiple Data Store locations should make use of this and always try to write new data only to healthy Data Store nodes. For Skyprobe, Data Store/Camera intends to provide a symbolic link in a higher level filesystem which will always point to the best choice Data Store location for new data.

4.2 Disk Maintenance and Replacement

Hardware RAID systems often feature background bad sector scans. These scans ensure that read errors are found early, before multiple ones have accumulated. Multiple read errors across different disk discovered at the same time are one of the main weaknesses of RAID-1 and RAID-5. While Linux Software RAID does not currently perform and background sector scans, we will implement a simple one which will run at a user configurable time. The duration of the scan, with the current generation of 400 GB disks, is approximately one hour. For the case of the summit pixel servers, we plan to schedule this for the middle of the day, when the load is expected to be lightest.

In addition to sector scans, the Data Store software will read the drives own SMART registers. These registers count various types of errors, and are used by the manufacturer to decide if the disk is suitable for replacement.

Starting with an array of 4 healthy disks, the Data Store internal system software will react to the following events:

1. Starting with 4 good disks in the RAID:
 - 1.a. Event: The SMART status of one drive goes bad
Action: Remove and light the FAIL LED
 - 1.b. Event: RAID itself ejects a disk (first time)
Action: Check read-write, check SMART, re-add.
 - 1.c. Event: RAID ejects the disk for 2nd time in 2 weeks.
Action: Remove and light the FAIL LED
 - 1.d. Event: Sector scan finds any kind of problem (first time)
Action: Check SMART, remove, re-add.
 - 1.e. Event: Sector scan finds problem 2nd time in 2 weeks.
Action: Remove and light the FAIL LED

2. Starting with a degraded RAID:
 - 2.a. Event: The SMART status of 1/3 good drives goes bad
Action: Make RAID read-only. Light system fault LED.
 - 2.b. Event: RAID itself craps out (another disk fails)
Action: Light system fault LED.
 - 2.c. Event: Sector scan wants to run
Action: Suppress it!
 - 2.d. Event: Drive hotplug detected
Action: Turn off red LED. Begin reconstructing the array.
 - 2.e. Event: Reconstruction fails
Action: Force cleanup daemon to remove all files which have expired now. Then fill disk with a file full of nulls. Then remove the file and re-try reconstruction.
 - 2.f. Event: Reconstruction fails again
Action: Light system fault LED.
(TBD What if the replacement disk was just the only problem?
If we can detect that, instead turn its FAIL LED on again?)
 - 2.g. Event: Reconstruction completes.
Action: Re-enable sector scans, reset system fault LED if needed.

If your system does not have the FAIL LEDs, you will have to rely on status of the Web page to determine which disk to replace and when.

5 Cost

While the Data Store software itself could utilize arbitrary filesystem space, the camera group's testing and intended use will be on XFS formatted (journalled) 4-disk RAID-5. Data store systems used by subprojects that have small amounts of data to exchange, and those which may be located off the summit, having nothing to do with the camera subsystem should still plan to purchase a complete hardware and software solution specified by the camera group. The subproject will be responsible for purchasing, monitoring (as described in the previous sections), and repairing the hardware. There are several reasons that we want this standard:

- Consistency of Data Stores across PS1.
- Security of services (NFS, HTTP download)
- Specified performance of the Data Store is heavily dependent on OS and hardware.
- Quota feature can be OS and filesystem dependent.
- Hardware monitoring (temperature, disk failures, etc.) is OS and hardware dependent.

In summary, although Data Store is primarily a software system, the features and specifications described here assume a standard for the supporting OS and hardware.

The current default 1U data store unit is a Tyan Barebone server with 4 SATA disks in a software RAID-5 configuration. Price varies depending on the amount of disk space required. The table lists the LOW and HIGH end of the spectrum that is possible today (September 2007). In cases where there is a similar application to the Pixel Servers, which can utilize processing power on a Linux platform (with local access to datastore data, without even using NFS) then the HIGH processing option may also be desirable.

CPU Cores	Storage	Case + M.board	Root flash	RAM	CPU	DISK	SHIP	TOTAL
LOW (4x1.6)	LOW (1.2TB)	700 (-LC version)	80	100(1G)	240	400	100	\$1620
HIGH (8x1.6)	MED (2.2TB)	830	80	170(2G)	480	840	100	\$2500(*)
HIGH (8x1.6)	HIGH(3TB)	830	80	170	480	1360	100	\$3020

(*)=This is the camera group Pixel Server configuration.

Separate Data Store nodes that cannot rely on the summit NFS root server, or do not want the dependency, will need the \$80 IDE-CF root disk option included in the table above. Camera group can supply the system flash image including all of the Data Store software and operating system, to go on a 4GB flash. Camera Pixel Servers do not include this, since they are network-booted (w/ DHCP and PXE).

Totals shown include shipping. Assembly is minimal (plugging in of disks and CPUs.) More information about our **OLD generation** of Supermicro servers can be found at <http://lolo.ifa.hawaii.edu:872/x6dal/>.