MONSOON
Data Handling System
Interface
Status and Data Stream Transfer

NOAO Document ICD 1.0
Revision: 1.0

Authored by:
Nick C. Buchholz, Greg Chisholm, Phil N. Daly and Peter Ruckle
1/7/2002
Please send comments:
nbuchholz@noao.edu
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date Approved</th>
<th>Sections Affected</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.1</td>
<td>3/11/2002</td>
<td>All</td>
<td>First release draft</td>
</tr>
<tr>
<td>0.0.2</td>
<td>4/12/2002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>8/8/2006</td>
<td>All</td>
<td>Reformat and edit - aro</td>
</tr>
</tbody>
</table>
Table of Contents

Revision History ................................................................................................................. 2
Table of Contents ............................................................................................................... 3
1.0 Introduction ............................................................................................................... 5
  1.1 Scope ...................................................................................................................... 5
  1.2 Acronyms and Glossary .......................................................................................... 5
    1.2.1 Abbreviations and Acronyms ............................................................................ 5
    1.2.2 Glossary ............................................................................................................. 6
2.0 DHS Requirements for Instrument Support .......................................................... 9
  2.1 Data Throughput Rates .......................................................................................... 9
  2.2 Reliability ............................................................................................................... 10
  2.3 Meta-Data Handling ............................................................................................. 10
  2.4 Data Storage/Archiving ....................................................................................... 11
  2.5 DHS Control Requirements ................................................................................. 11
  2.6 Use-Case Handling ............................................................................................. 12
  2.7 Failure Mode Handling ....................................................................................... 12
3.0 DHS Application Programmer Interface .................................................................. 12
  3.1 Standard Calling Sequence for DHS API Functions ............................................. 12
  3.2 dhsSysOpen – To DHS Supervisor ....................................................................... 13
  3.3 dhsOpenConnect – To DHS Node ......................................................................... 13
  3.4 dhsOpenExp – To DHS Node ................................................................................. 14
  3.5 dhsSysClose – To DHS Supervisor ....................................................................... 15
  3.6 dhsCloseDisconnect – To DHS Node .................................................................... 15
  3.7 dhsCloseExp – To DHS Node ................................................................................. 16
  3.8 dhsSendMetaData – To DHS Supervisor ............................................................... 17
  3.9 dhsSendPixelData – To DHS Node ....................................................................... 18
  3.10 dhsReadImage - To DHS Node ............................................................................ 19
  3.11 dhsIOCtl – To DHS Node ..................................................................................... 19
  3.12 DHS API Data Types and Structure Definitions .................................................. 21
  3.13 DHS Parser – What Does the DHS Know and How Does it Know .................... 24
  3.14 DHS API Behavior in the Face of DHS Failures .................................................... 24
  3.15
  3.16 DHS Communications Internals .......................................................................... 24
  3.17 Byte Order ........................................................................................................... 24
  3.18 Message Structure ............................................................................................... 25
  3.19 DHS Interface Message Types ............................................................................. 25
Appendix I DHS Requirements for NEWFIRM Instrument Support .................................. 26
Appendix II DHS API Function Internals ....................................................................... 26
Appendix III Attributes Produced by a GPX System ...................................................... 27
List of Figures

Figure 1 - Observatory System Reference Model ........................................................ 8
Figure 2 - System Context Diagram ........................................................................... 9

List of Tables

Table 1 - Attributes ..................................................................................................... 27
1.0 Introduction

1.1 Scope

The Data and Status outputs from the GPX (ICD 4.0) have so far been left unspecified. It has been agreed to some extent that a system for publishing the data and status is required for all instruments. This document serves as a beginning of the discussion of the interface to the data and status streams from instruments.

In addition, while developing the ICD 4.0 description of the Data and Status outputs, the conclusion was reached that an ICD describing what the DHS Interface can expect to receive from the entire Observation Control System was more appropriate. It is assumed that the GPX will conform to this ICD completely.

The document is divided into the following sections:

- **Section 1.0 - Introduction.**
- **Section 2.0 - DHS Requirements**
- **Section 3.0 - DHS Application Programmer Interface**
- **Section 3.16 - DHS Communications Internals**

The intended audience for this document is:

Anyone with an interest in the Data Handling System Interface.

1.2 Acronyms and Glossary

1.2.1 Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Acquisition Camera</td>
</tr>
<tr>
<td>DCS</td>
<td>Detector Controller System (software)</td>
</tr>
<tr>
<td>DHE</td>
<td>Detector Head Electronics</td>
</tr>
<tr>
<td>DHS</td>
<td>Data Handling System</td>
</tr>
<tr>
<td>FITS</td>
<td>Flexible Image Transport System</td>
</tr>
<tr>
<td>FP</td>
<td>Focal Plane</td>
</tr>
<tr>
<td>FPA</td>
<td>Focal Plane Array</td>
</tr>
<tr>
<td>GPX</td>
<td>Generic Pixel Server</td>
</tr>
<tr>
<td>IAS</td>
<td>Image Analysis System</td>
</tr>
<tr>
<td>ICS</td>
<td>Instrument Control System</td>
</tr>
<tr>
<td>IDPS</td>
<td>Image Data Preprocessor System</td>
</tr>
<tr>
<td>ID</td>
<td>Identifier</td>
</tr>
<tr>
<td>IR</td>
<td>Infrared</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>N/A</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>MSL</td>
<td>MONSOON Supervisory Layer</td>
</tr>
</tbody>
</table>
1.2.1 Abbreviations and Acronyms (Cont.)

MONSOON        Not an acronym
NOCS           NEWFIRM Observation Control System
OCS            Observatory Control System
SUS            Status Update System
TBD            To Be Decided

1.2.2 Glossary

**Attribute** - An entity which describes some aspect of the configuration of an Pixel Server System, such as the Level of a voltage or the state of a shutter. Some attributes will be used by the Pixel Server System as command parameters. The OCS communicates with a science instrument by sending it sets of “attributes” and “values”.

**Command** - An instruction commanding a system to start some action. The action may result in a voltage changing or some internal parameters being set to particular values. A command may have command parameters (aka. “arguments”) which contain the details of the instruction to be obeyed.

**Pixel Acquisition Node** - A component of a Generic Pixel Server, this is the computer which handles the interface to the detector head electronics and the image pre-processing of the data stream from the detector head electronics.

**Data Array** - The data, while it is stored in data processing memory, which resulted from one or more readouts of an IR array or CCD detector.

**Data Set** - A self-contained collection of data generated as a result of an Pixel Server obeying a gpxStartExp command. Each gpxStartExp command results in one and only one data set.

**Exposure** - The name used to describe the process and the data resulting from the process of resetting/clearing a detector, exposing it to photons and then reading one or more frames to determine the photon levels. These frames are processed into a data array, called an exposure, which may be further processed. (For example, an exposure would be the data array which results when a single Reset-Readout-Integrate-Readout cycle is performed on an IR detector or a single CCD Clear-Integrate-Readout cycle.)

**Exposure Sequence** - The process by which valid data is produced. Various levels of exposure sequencing occur during an observing run. At the lowest level there are the Reset-Readout-Integrate-Readout or Clear-Integrate-Readout cycles that result in a single IR or OUV exposure. At the highest level are the observing sequences that move the telescope, configure the instrument and take a series of exposures that create an observation.

**Frame** - A frame is the result of one or more readouts of an array averaged pixel by pixel. Each frame represents the signal values obtained from reading the entire ROI being read out of the detector. Multiple frames may be processed into a single exposure.
1.2.2 Glossary (Cont.)

**Image** - The array of detector pixel and description data representing a science or diagnostic image or spectrum. An image is capable of being displayed or processed as a discrete entity. The values in the array may be stored in memory or on disk and are related to the data taken by the detector by some processing algorithm, (for example an image may consist of all the coadded and averaged exposures in one beam of a chop mode gpxStartExp command).

**Observation** - The process of exposing the detector to photons in one or more exposures. The result of an observation is a picture.

**Read** - When used as a noun to describe instrument data, this refers to a single read of a pixel on the detector. A read may consist of several A/D conversions of the pixel data that are averaged or processed in some other way to produce a single integer output value for the pixel. A readout is made up of one read of each pixel in the detector ROI being read.

**Readout** - When used as a noun to describe instrument data, this refers to a single read of every pixel in the detector. One or more readouts can be averaged pixel by pixel to create a frame.

**Region of Interest** - A sub array of the available detector area. There are two types of sub arrays that can be defined. The Sequence ROI is on the active surface of the array used to increase the frequency of the Array readout. The Data Reduction ROI is an arbitrary rectangle of any size which fits on the Array. Data Reduction ROIs are defined to reduce the volume of data sent to the disk or DHS even when the entire Array is being read out.

**Value** - The value associated with an “attribute”.

**Detector Head Electronics** - The lowest level hardware system, normally closely connected to the detector and the dewar in which the detector resides.

**Pixel Acquisition Node** - The computer which handles the interface to the detector head electronics and the image pre-processing of the data stream from the Detector Head Electronics.

**Pixel Server System** - The combination of the Detector Head Electronics and a Pixel Acquisition Node which are coordinating the task of taking exposures and archive the resulting data set

**Pixel Server** - A system which produces pixel values when requested to do so by some client system.

**Generic Pixel Server** - A pixel server that conforms to the GPX Interface description.

**Supervisory Node** - A computer capable of controlling multiple Image Acquisition systems. The computer which runs the software which conforms to the GPS interface.
Reference Documents

SPE-C-G0037, “Software Design Description”, Gemini 8m Telescopes Project.

“ICD/16 — The Parameter Definition Format”, Steve Wampler, Gemini 8m Telescopes Project.

WHT-PDF-1, “FITS headers for WHT FITS tapes”, Steve Unger, Guy Rixon & Frank Gribbin, RGO.

NOST 100-1.0, “Definition of the Flexible Image Transport System (FITS)”, NASA Office of Standards and Technology.

GEN-SPE-ESO-00000-794, “ESO Data Interface Control Document”, Miguel Albrecht, ESO.


ANSI/IEEE Std 754-1985 - “IEEE Standard for Binary Floating-point Arithmetic” - Standards Committee of the IEEE Computer Society, USA 19850812

xxxx “XDR - Extended Data Representation Standard” ????

ICD 6.0 - “Generic Detector Controller - Command and Data Stream Interface Description”, Nick C. Buchholz (NOAO), Barry M. Starr (NOAO), Version 0.1.2, dated: 200203011- NOAO Document Number MNSN-AD-01-0004

---

Observatory System Reference Model

Figure 1
2.0 DHS Requirements for Instrument Support

2.1 Data Throughput Rates

The data transfer mechanism should be capable of meeting the following throughputs. Note that if large numbers of Fowler Samples are done the rate at which data can be transferred can be significantly reduced due to data generation priority resulting in lower long-term throughputs.

2.1.1 Normal Data Rate

Normal operation will be no more than one 80Mb frame every second.

2.1.2 Maximum Data Rate

High rate, short duration bursts may result in one 80Mb data frame per second. This will result in an 80-MB image data frame and two ~128 KB meta-data buffers per frame.

2.1.3 Maximum Attainable Data Rate Used

The data transfer mechanism should transfer at the maximum data rate that can be attained. The transfers will be tuned for large block transfers.
2.1.4 Graceful Degradation in Face of Failures

The DHS should be capable of continuing to transfer and archive data even in the face of failure of one or more DHS machines. Observing should be able to continue, but at a considerably slower rate, even if all DHS computers have failed. The DHS should be able to write FITS images to the local PAN disk if no DHS computers are operating.

2.2 Reliability

2.2.1 Safe Store Transport

The data transfer will complete and correctness/completeness will be checked before the OK response is returned to the data source. The data will be safe if stored to disk as soon as possible.

2.2.2 Data Completeness

Data transfers must result in data on the destination being complete, that is, all values at the source are transferred to the destination.

2.2.3 Data Correctness

Data transfers must result in data on the destination having the same values as data on the source.

2.2.4 DHE Data Collection Priority

Data transfers from the DHE to the PAN must have priority over all other PAN operations. The nature of the DHE hardware requires prompt handling of the incoming data or pixel data will be lost. If necessary, the DHS protocols will have the capability of being blocked (on a semaphore or other mechanism) when the DHE data transfer starts.

2.3 Meta-Data Handling

2.3.1 Multiple Meta-Data Sources

The DHS must be able to collect meta-data from multiple sources.

2.3.2 Multiple Meta-Data Formats

The DHS must be able to collect meta-data in multiple formats. The API must provide a mechanism for describing the format of the arriving meta-data.

2.3.2.1 Attribute-Value-Comment Triplets
2.3.2.2 Multi-dimensional Arrays of a Data Type
2.3.2.3 ROI Image Arrays - Multiple Focus Frames in Single Image
2.3.2.4 Shift Lists – Groups of XY Shift Values
2.3.2.5 Guide Image Arrays – Multiple Guide Frame Groups in a Single Image
2.3.2.6 Multi-dimensional Arrays with Mixed Data Types
2.3.3 Single and Block Meta-Data Transfers
The DHS must be able to collect meta-data as individual attribute-value pairs or as blocks of such attribute-value pairs.

2.3.4 Combine/Associate Meta-Data and Pixel Data
The DHS must be able to combine/associate the meta-data and the pixel data that are generated by a single exposure. This association will be by exposure id provided with each piece of data sent to the DHS.

2.4 Data Storage/Archiving

2.4.1 Memory to Ethernet Transfers
Normal data transfers will be from memory to memory with a minimum of data copying. Ideally data will only be copied once, from the MONSOON Buffer to the Ethernet or other transport hardware.

2.4.2 Save Every Frame to Disk
The system must be capable of saving every generated frame to disk or other permanent storage medium for later analysis.

2.4.3 Combine/Associate Image Sections from Multiple Sources (PANs)
The DHS must be able to accept images in several pieces and combine or associate the pieces that constitute a single exposure

2.4.4 Combine/Associate Multiple Exposure Types (Observations).
The DHS must be able to accept multiple exposures of a variety of types and sizes and combine them into a single Observation Set

2.5 DHS Control Requirements

2.5.1 Regularized
The control of the DHS from the data sources will be completely through the programmer’s API described in Section 3.0

2.5.2 Debugging Support
The DHS interface library will provide for several levels of feedback to the PAN software.
2.5.3 Simulation Support
The DHS interface library will provide for a simulation mode that does not connect to the DHS machines. All data will be discarded in this mode without accessing any machine off the PAN.

2.6 Use-Case Handling

2.7 Failure Mode Handling

3.0 DHS Application Programmer Interface
The DHS interface API will be included entirely within a single shared library loaded at runtime into the data publication processes of an instrument. The library may use whatever internal data transfer and connection methods the DHS group deems appropriate. The nature of these protocols will be buried within the DHS API library. It is possible that some of these functions will be NULL operations for certain implementations of the DHS Library based on the underlying protocol. For instance, the local disk FITS writer included with the basic MONSOON software does not use any information from the dhsSysOpen or dhsOpenConnect calls. As a result these functions are No-Ops.

3.1 Standard Calling Sequence for DHS API Functions
Each DHS API function will have a defined parameter set as follows:

dhsXxxYxxxx( long *status, char *response, dhsHandle dhsId, { additional parameters} );

3.1.1 Parameter 0 - long *status
This parameter performs two functions. First, it passes into the routine the status value generated by earlier routines in a sequence. Second, it provides a means for each routine to report its status to following routines. The first step in each routine should be a check of the incoming status value and a return if the status indicates a previous error.

3.1.2 Parameter 1 - char *response
This parameter is a character buffer pointer that provides a means for status information to be returned to the upper levels of the process and eventually the user. The pointer may be null. However, if it is non-null, it must point to a memory buffer that contains at least 4096 bytes. Each routine should detect the presence of this address and fill in the buffer only if the pointer is non-null.

3.1.3 Parameter 2 - dhsHandle dhsId
Each DHS routine will pass in a handle parameter obtained by doing a dhsSysOpen or dhsOpenConnect. The dhsSysOpen and dhsOpenConnect routines will pass a dhsHandle pointer (dhsHandle *). These routines will fill in the handle pointer with a successfully opened DHS handle or, in case of a failure, an ERROR(-1) value.
3.2 dhsSysOpen - To DHS Supervisor

This function is used by the data generator control processes (NOCS and MSL in NEWFIRM) to open a connection to the DHS supervisor node. These connections are used to send the supervisor information about the overall configuration of the observation and focal plane. The function may perform some configuration of the DHS library and DHS supervisor.

3.2.1 Calling Sequence

This function must be called before any system can begin configuring the DHS or sending or requesting image or meta-data from the DHS. The function call takes the form:

\[ \text{dhsSysOpen (long *status, char *response, dhsHandle *dhsID, ulong whoAmI)} \]

3.2.2 Additional Parameters

ulong - whoAmI - an Identifier that determines the identity of the data generator making the call (an MSL or a NOCS in NEWFIRM). \( \text{(This may not be needed and may be eliminated in future versions.)} \)

3.2.3 Returns

In \textit{status} the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention, any non-negative number indicates success.

In \textit{response} the function returns a text message that gives the status of the function call.

In \textit{dhsID} the function returns a handle to be used in all future DHS library function calls from this process. If the function call fails the \textit{dhsID} field contains ERROR (-1) on return.

3.2.4 Function Internals

See Appendix II, DHS/API Function Internals.

3.3 dhsOpenConnect - To DHS Node

This function is used by a data generator process to open a connection to and perform the initial configuration of a DHS node. The DHS node may be a separate machine or process within the DHS. This connection is then used to send configuration, pixel and meta-data to the DHS node.

3.3.1 Calling Sequence

This function must be called before any data generator sub-system can begin communicating with a DHS node. The call assumes that the major system has previous made a \textit{dhsSysOpen} call and has done the required configuration of the DHS to get ready for connections from the sub-systems. The function call takes the form:

\[ \text{dhsOpenConnect (long *status, char *response, dhsHandle *dhsID, ulong whoAmI,}
\text{ struct fpConfig *pxlConfig)} \]
3.3.2 Additional Parameters

ulong - *whoAmI* - an Identifier which determines whether the call is being made by PAN or by another entity sending data to the DHS. *(This may not be needed and may be eliminated in future versions.)*

struct fpConfig * - *pxlConfig* - a pointer to a focal plane configuration structure which describes the portion and position of the focal plane which will be affected by this connection. *(See Section 3.12.3.)*

3.3.3 Returns

In *status* the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention, any non-negative number indicates success.

In *response* the function returns a text message that gives the status of the function call.

In *dhsID* the function returns a handle to be used in all future DHS library function calls from this process. If the function call fails the *dhsID* field contains ERROR (-1) on return.

3.3.4 Function Internals

See Appendix II, DHS/API Function Internals.

3.4 dhsOpenExp - To DHS Node

This function is used by a data generator process to open an exposure connection and perform the configuration of an exposure transfer to a DHS node. The DHS node may be a separate machine or process within the DHS. This connection is then used to send pixel and meta-data to the DHS node.

3.4.1 Calling Sequence

This function must be called before any sub-system can begin sending exposure pixel and/or meta-data to a DHS node. The call assumes that the sub-system has previously made a *dhsOpenConnect* call and has done the required configuration of the DHS node to get ready for exposure pixel and meta-data from the sub-system. The function call takes the form:

```c
    dhsOpenExp (long *status, char *response, dhsHandle dhsID, struct fpConfig *pxlConfig,
                double *expID, char *obsSetID)
```

3.4.2 Additional Parameters

struct fpConfig * - *pxlConfig* - a pointer to a focal plane configuration structure which describes the portion and position of the focal plane that will be affected by this connection.

double - *expID* - a pointer to a unique double value derived from the observation control system and associated with a single exposure (the Monsoon Star Date). All data sent to the DHS without an associated exposure ID should be tagged with this exposure ID. If this value is NULL each piece of pixel or meta-data sent to the DHS must be tagged with the exposure ID of the associated exposure.

char * - *obsSetID* - a pointer to a string observation data set ID derived from the observation control system and associated with a set of exposures which make up an observation. All data sent to the DHS without an associated exposure ID should be tagged with this exposure ID. If this value is NULL, each piece of pixel or meta-data sent to the DHS must be tagged with the observation Set ID of the associated exposure.
3.4.3 Returns
In status the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.

In response the function returns a text message that gives the status of the function call.

3.4.4 Function Internals
See Appendix I, DHS/API Function Internals.

3.5 dhsSysClose - To DHS Supervisor
This function is used by a data generator control process (NOCS and MSL in NEWFIRM) to close a connection to the DHS supervisor. When this routine is called it is assumed that no new observations or exposures will be initiated and when all current observations and exposures are complete, the DHS can shut down.

3.5.1 Calling Sequence
This function should be called after all observations for a session are complete. The data generator control processes (NOCS and MSL in NEWFIRM) will shut down after this call. The function call takes the form:

dhsSysClose (long *status, char *response, dhsHandle dhsID)

3.5.2 Additional Parameters
None

3.5.3 Returns
In status the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.

In response the function returns a text message that gives the status of the function call.

3.5.4 Function Internals
See Appendix II, DHS/API Function Internals.

3.6 dhsCloseConnect - To DHS Node
This function is used by a data generator control process to close a connection to a DHS node. When this routine is called, it is assumed that no new observations or exposures will be initiated and that any current observations and exposures that are incomplete will not be completed. The DHS node can perform whatever cleanup is required and shut down.

3.6.1 Calling Sequence
This function should be called after all exposures for a session are complete. The process making this call will shut down after this call. The function call takes the form:

dhsCloseConnect (long *status, char *response, dhsHandle dhsID)
3.6.2 Additional Parameters
None

3.6.3 Returns
In \textit{status} the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.

In \textit{response} the function returns a text message that gives the status of the function call.

3.6.4 Function Internals
See Appendix II, DHS/API Function Internals.

3.7 \texttt{dhsCloseExp} - To DHS Node
This function is used by a data generator control process to close an exposure on a DHS node. When this routine is called it is assumed that no additional exposure pixel or meta-data will be sent for the closed exposure and that any current exposures that are not complete will not be completed. The DHS node can perform whatever cleanup is required for an exposure and prepare for the next exposure.

3.7.1 Calling Sequence
This function should be called after all data for an exposure have been sent. The process making this call will send no additional data associated with this exposure after this call. The function call takes the form:
\begin{verbatim}
dhsCloseExp (long *status, char *response, dhsHandle dhsID, double expID)
\end{verbatim}

3.7.2 Additional Parameters
double - expID - a unique double value derived from the observation control system and associated with a single exposure (the Monsoon Star Date).

3.7.3 Returns
In \textit{status} the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.

In \textit{response} the function returns a text message that gives the status of the function call.

3.7.4 Function Internals
See Appendix II, DHS/API Function Internals.
3.8 dhsSendMetaData- To DHS Node

This function is used by a data generator control process to send meta-data to a DHS node. When this routine is called the meta-data to be sent is assumed to be formatted according to the current mdConfig structure that was set by a dhsOpenExp or a subsequent dhsIOCtl function MD_CONFIG call.

3.8.1 Calling Sequence

This function should be called to send meta-data for an open exposure to a DHS node. The process making the call will take care of freeing any memory and any additional cleanup after the call is complete. The call should not return until all meta-data in the memory block has been received and confirmed on the DHS node. The function call takes the form:

\[
dhsSendMetaData (\text{long } *\text{status}, \text{char } *\text{response}, \text{dhsHandle } \text{dhsID}, \text{void } *\text{blkAddr}, \text{size}_t \text{ blkSize,})
\]
\[
\text{struct mdConfig } *\text{mdDescrptr, double } *\text{expID}, \text{char } *\text{obsSetID})
\]

3.8.2 Additional Parameters

void * - blkAddr - The address of the formatted meta-data in memory.

size_t - blkSize - the size of the memory block containing the meta-data. The blkSize and the current mdConfig will allow the calculation of the number of meta-data items in the block.

struct mdConfig * - mdDescrptr - a pointer to a meta-data descriptor block describing the structure of the current meta-data block.

double * - expID - a pointer to a unique double value derived from the observation control system and associated with a single exposure. (the Monsoon Star Date). If this value is NULL, the meta-data sent to the DHS will be tagged with the default exposure ID assigned by the dhsOpenExp. If no default was assigned, the function should return an error.

char * - obsSetID - a pointer to a string observation data set ID derived from the observation control system and associated with a set of exposures which make up an observation. If this value is NULL, the meta-data sent to the DHS will be tagged with the observation Set ID assigned by the dhsOpenExp. If no default was assigned, the function should return an error.

3.8.3 Returns

In status the function returns OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.

In response the function returns a text message that gives the status of the function call.

3.8.4 Function Internals

See Appendix II, DHS/API Function Internals.
3.9 **dhsSendPixelData - To DHS Node**

This function is used by a data generator control process to send pixel data to a DHS node. When this routine is called, the pixel data to be sent is assumed to be formatted as an image starting at the lower left corner of the rectangle and continuing in row major order to the upper right. The pixel size (data type), location and number of rows and columns in the data are described by the fpConfig structure in the call.

3.9.1 **Calling Sequence**

This function should be called to send pixel data for an open exposure to a DHS node. The process making the call will take care of freeing any memory and any additional cleanup after the call is complete. The call should not return until all pixel data in the memory block has been received and confirmed on the DHS node. The function call takes the form:

```
dhsSendPixelData (long *status, char *response, dhsHandle dhsID, void *pxlAddr, size_t blkSize,
                 struct fpConfig *pxlDescrptr, double *expID, char *obsSetID)
```

3.9.2 **Additional Parameters**

- **void * - pxlAddr** - The address of the formatted meta-data in memory.
- **size_t - blkSize** - the size of the memory block containing the meta-data. The blkSize and the current mdConfig will allow the calculation of the number of meta-data items in the block.
- **struct fpConfig * - pxlDescrptr** - a pointer to a focal plane descriptor block describing the structure of the current block of pixels and their location in the focal plane.
- **double - *expID** - a pointer to a unique double value derived from the observation control system and associated with a single exposure (the Monsoon Star Date). If this value is NULL, the pixel data sent to the DHS will be tagged with the default exposure ID assigned by the dhsOpenExp. If no default was assigned the function should return an error.
- **char * - obsSetID** - a pointer to a string observation data set ID derived from the observation control system and associated with a set of exposures which make up an observation. If this value is NULL the pixel data sent to the DHS will be tagged with the observation Set ID assigned by the dhsOpenExp. If no default was assigned the function should return an error.

3.9.3 **Returns**

In *status* the function returns **OK, SIMULATION_OK, a positive number information code or a negative number error code. By convention any non-negative number indicates success.**

In *response* the function returns a text message that gives the status of the function call.

3.9.4 **Function Internals**

See Appendix II, DHS/API Function Internals.
3.10 **dhsReadImage** - To DHS Node

TBD

3.11 **dhsIOCtl** - To DHS Node

dhs IOCtl (long *status, char *response, dhsHandle dhsID, ulong ioCtlFunction,
char *ObsID, double ExpID, ... { a parameter list of additional parameters})

3.11.1 **dhsIOCtl Function - Observation Configuration - OBS_CONFIG**

This function will normally be performed by the data generator control processes (NOCS or MSL). It assigns an ObsSetID to an observation configuration that will be used to determine the size and constituents of the observation.

3.11.1.1 Parameter List

- struct obsSetConfig *obsConfig -

3.11.1.2 Returns

3.11.2 **dhsIOCtl Function - Meta-Data Configuration - MD_CONFIG**

This function can be used to set the default structure of future meta-data to be transferred to the DHS. (This is a redundant function. A dhsCloseExp followed by a dhsOpenExp with a new configuration could be used instead.)

3.11.2.1 Parameter List

- struct mdConfig *defMDConfig -

3.11.2.2 Returns

3.11.3 **dhsIOCtl Function - Image Configuration - FP_CONFIG**

This function can be used to set the default focal plane configuration to be used in transferring pixel data to the DHS. (This is a redundant function. A dhsCloseConnect followed by a dhsOpenConnect with a new focal plane configuration could be used instead.)

3.11.3.1 Parameter List

- ulong whoAmI - an unsigned long specifying what level of the system is sending the command. If it is the MSL, the DHS supervisor would have to be informed of the change. (It is unlikely this will be needed in normal circumstances.)

- struct fpConfig *pxlConfig -

3.11.3.2 Returns

3.11.4 **dhsIOCtl Function - Keyword Translation Configuration - KEYWORD_TRANS**

3.11.5 **dhsIOCtl Function - Set Debug Level - DEBUG_LVL**

3.11.6 **dhsIOCtl -Function - Set Simulation Level – SIMULATION**

3.11.7 **dhsIOCtl-Function – Set FITS Image Destination Directory -**
3.11.8  dhsIOCtl -Function – Set FITS image file base name.

3.11.9  dhsIOCtl -Function – Set FITS image file index.

3.11.10 dhsIOCtl -Function – Set DHS supervisor Machine Name

3.11.11 dhsIOCtl -Function – Set DHS Node Machine ID – IP or Name
3.12 DHS API Data Types and Structure Definitions.

3.12.1 Exposure Identifier

Each exposure generated by the NOCS and MONSOON GPX, as well as all of the meta-data associated with that exposure, will be identified with a unique identifier. The msd (MONSOON Star Date) is a ‘C’ double value based on the Julian Date and time. The msd is assigned to an exposure by the NOCS and will be sent to identify all data sent to the DHS with that exposure.

3.12.2 Observation Set Identifier

An Observation set is a series of exposures; sky flats, object images, darks, and calibration images that will be processed together to create a science observation. Each exposure taken within an observation set will be identified by a unique observation set identifier. At this time, this identifier consists of a 128-character string appended with the msd of first observation.

3.12.3 Focal Plane Configuration Structure

- This structure is used at several levels to describe the pixel/focal plane configuration.
- At the MSL level it gives the size shape and make up of the entire focal plane.
- At the PAN, panSaver, level it gives the portion of the focal plane being handled by this PAN.
- At the fSaver process level, in a dhsSendPixelData call, it describes the portion of the focal plane being sent by this call.

The ‘C’ code definition of the fpConfig structure is:

```c
struct fpConfig {
    ulong xSize;  /* the Size of a row on the focal plane, Number of Columns*/
    ulong ySize  /* the Size of a column on the focal plane, Number of Rows*/
    ulong xStart; /* the column index of the first pixel in this portion of the focal plane*/
    ulong yStart /* the row index of the first pixel in this portion of the focal plane */
    long dataType /* the data type of the pixels to be or being sent,Number of Bytes */
};
```

xSize - this field gives the size of a row in the pixel data. Used By the MSL it is the total number of columns in the focal plane. Used by panSaver it is the number of columns handled by this PAN. Used in a dhsSendPixelData call it is the number of columns being sent by this call.

ySize - this field gives the size of a column in the pixel data. Used By the MSL it is the total number of rows in the focal plane. Used by panSaver it is the number of rows handled by this PAN. Used in a dhsSendPixelData call it is the number of rows being sent by this call.
xStart - this field gives the column index of the first pixel in this portion of the focal plane. Indices run from 1 to n where n is the total number of columns. Used by the MSL, this should always be 1. Used by panSaver, it will be the column index of the first pixel in the lower left corner of the focal plane portion on this PAN. Used in a dhsSendPixelData call, it is the column index of the first pixel in the lower left corner of the portion of the focal plane being sent by this call.

yStart - this field gives the row index of the first pixel in this portion of the focal plane. Indices run from 1 to n where n is the total number of rows. Used by the MSL, this should always be 1. Used by panSaver, it will be the row index of the first pixel in the lower left corner of the focal plane portion on this PAN. Used in a dhsSendPixelData call, it is the row index of the first pixel in the lower left corner of the portion of the focal plane being sent by this call.

dataType - this field gives a value representing the data type of the pixels being sent. The field should be filled in by one of the following symbolic constants:

- UBYTE - each pixel is a value between 0 and 255 (0x00-0xFF) and occupies one byte in memory
- BYTE - each pixel is a value between -128 and 127 (0x80-0x7F) and occupies one byte in memory
- USHORT - each pixel is a value between 0 and 65535 (0x0000-0xFFFF) and occupies two bytes in memory
- SHORT - each pixel is a value between -32768 and 32767 (0x8000-0x7FFF) and occupies two bytes in memory
- INTEGER - each pixel is a value between 0 and 4294967295 (0x00000000-0xFFFFFFFF) and occupies four bytes in memory
- UNSIGNED - each pixel is a value between -2147483648 and 2147483647 (0x80000000-0x7FFFFFFF) and occupies four bytes in memory
- VLONG - each pixel is a value between 0 and \(2^{64}-1\) (0x0000000000000000-0xFFFFFFFFFFFFFFFFFF) and occupies eight bytes in memory
- UVLONG - each pixel is a value between \(-1\times2^{63}\) and \(2^{63}-1\) (0x8000000000000000-0x7FFFFFFF00000000) and occupies eight bytes in memory
- FLOAT - each pixel is a value between \(-xxx.zzz \times 10^{yy}\) and \(xxx.zzz \times 10^{yy}\) and occupies four bytes in memory
- DOUBLE - each pixel is a value between \(-xxx.zzz \times 10^{yy}\) and \(xxx.zzz \times 10^{yy}\) and occupies eight bytes in memory

3.12.4 Meta-Data Descriptor structure

This structure is used to describe the configuration of the meta-data being sent to the DHS. Processes that call the dhsSendMetaData routine can fill in this structure to describe the meta-data being sent from this process by this call.
The ‘C’ code definition of the mdConfig structure is:

```c
struct mdConfig {
    ulong metaType;    /* the conceptual type of the meta data */
    ulong numfields;  /* the number of fields in the meta-data array */
    ulong fldSize[MAXFIELDS];  /* the number of items in the field, */
    ulong dataType[MAXFIELDS]; /* the data type of the data values in the field */
};
```

`metaType` - This field gives the type of the meta-data being sent. At this time the available types are:

- FITSHEADER - three fields, an 8-character name, a 32-character value field and a 40-character comment field.
- AVPAIR - three fields, a 64-character name a 64-character Value field and a 128-character comment field
- SHIFTLIST - 64 fields - each containing 2 UBYTES representing the x and y shifts given in one cycle of an OTA readout, centroid, shift routine.
- ARRAYDATA - N+2 fields which can be used to create an arbitrary N-dimensional homogenous binary array, fldSize[0] contains the number of dimensions N, fldSize[1] through fldSize[N] give the size of each dimension, fldSize[N+1] gives the total number of items in the array. dataType[0] through dataType[N] should be UBYTE, USHORT or UNSIGNED, dataType[N+1] should contain the datatype of the underlying array.

`numFields` - this field gives the number of fields in the meta-data. This must be less than MAXFIELDS.

fldSize[xx] - these fields give the number of items in each field. A -1 indicates the field is unused.

dataType[xx] - these fields give the data type of each field in the meta-data being sent. The field should be filled in by one of the following symbolic constants:

- UBYTE - each pixel is a value between 0 and 255 (0x00-0xFF) and occupies one byte in memory
- BYTE - each pixel is a value between -128 and 127 (0x80-0x7F) and occupies one byte in memory
- USHORT - each pixel is a value between 0 and 65535 (0x0000-0xFFFF) and occupies two bytes in memory
- SHORT - each pixel is a value between -32768 and 32767 (0x8000-0x7FFF) and occupies two bytes in memory
- INTEGER - each pixel is a value between 0 and 4294967295 (0x00000000-0xFFFFFFFF) and occupies four bytes in memory
- UNSIGNED - each pixel is a value between -2147483648 and 2147483647 (0x80000000-0x7FFFFFFF) and occupies four bytes in memory
- VLONG - each pixel is a value between 0 and \(2^{64} - 1\) (0x0000000000000000-0xFFFFFFFFFFFFFFFF) and occupies eight bytes in memory
- UVLONG - each pixel is a value between \(-1 \times (2^{63})\) and \((2^{63}-1)\) (0x8000000000000000-0x7FFFFFFF00000000) and occupies eight bytes in memory
- FLOAT - each pixel is a value between \(-xxx.zzz \times 10^{yy}\) and \(xxx.zzz \times 10^{yy}\) and occupies four bytes in memory
- DOUBLE - each pixel is a value between \(-xxx.zzz \times 10^{yy}\) and \(xxx.zzz \times 10^{yy}\) and occupies eight bytes in memory

For example, a fits header line might be described as numfields=3, fldSize = \{8,32,40\}, dataType = \{UBYTE, UBYTE,UBYTE\}

3.12.5 Array Data Descriptor Structure
3.12.6 Pixel Data Descriptor Structure
3.12.7

3.13 DHS Parser - What Does the DHS Know and How Does it Know?
This section will describe how the DHS will translate the meta-data into FITS headers or whatever to allow it to do something reasonable with the data.

3.13.1 Ascii Tables - FITS Header Data Translations
3.13.2 Ascii Tables - Attribute-Value Pair Data Translations
3.13.3 Binary Arrays - Like OTA Shift Lists

3.14 DHS API Behavior in the Face of DHS Failures

3.15
The remainder of this document will depend on implementation details of the DHS and libdhsUtil.so

3.16 DHS Communications Internals
This stuff may or may not belong in this document.

3.17 Byte Order

RULE
All messages SHALL use network byte ordering. Bytes in a message are labelled from 0 to N (where N is the length of the message).

3.17.1 Creating Messages

RECOMMENDATION
The sending software SHOULD build the message using htonl() or htons() or similar routines to convert from host ordering to network ordering. The receiving software SHOULD convert the message byte order into something which is usable locally using ntohl() or ntohs() or similar functions.
3.17.2 Strings

Strings that are embedded in a message SHALL BE inserted with the left-most character of the string in the lowest order message byte. For example, string “ABCD” will appear in message Byte N through N+3 with ‘A’ in byte n and ‘D’ in byte n+3.

3.17.3 Very Long Integers

The protocol implementers SHALL make provision for 64-bit integers by using the network ordering decision used for long integers. In the messages the Most Significant Byte of a very long integer SHALL BE sent first. That is, it will be closest to the start of the message.

3.17.4 Floating Point and Double Values

Real numbers represented by floats or doubles in ‘C’ SHALL BE represented in the messages in IEEE floating point format (see Bold Default Font). The byte ordering SHALL BE as defined in the XDR/Network standard.

3.18 Message Structure

DHS Interface messages SHALL have the following structure:

3.19 DHS Interface Message Types

3.19.1 Control Messages

3.19.2 Command/Response Communications Stream Definition

3.19.3 Status and Data Stream Interface

3.19.4 Status and Data Stream Interface
Appendix I  DHS Requirements for NEWFIRM Instrument Support

I.1  Data Throughput Rates
The data transfer mechanism should be capable of meeting the following throughputs. Note that if large numbers of Fowler Samples are done the rate at which data can be transferred can be significantly reduced due to data generation priority resulting in lower long-term throughputs.

I.1.1  Normal Data Rate
Normal operation will be no more than one 64-Mb frame every three seconds. (Assuming 4 digital averages and One Fowler sample)

I.1.2  Maximum Data Rate
High rate, short duration bursts may result in one 64-Mb data frame per second. This will result in a 64-MB image data frame and ~128 KB meta-data per frame.

I.1.3  Maximum Attainable Data Rate Used
The data transfer mechanism should transfer at the maximum data rate that can be attained. The transfers will be tuned for large block transfers.

Appendix II  DHS API Function Intervals
# Appendix III  Attributes Produced by a GPX System

## Table 1 - Attributes

<table>
<thead>
<tr>
<th>Keyword Name</th>
<th>Usage/Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>numArrays</td>
<td>An integer value giving the number of arrays in the focal plane.</td>
</tr>
<tr>
<td>arrayDescriptor</td>
<td>A structure describing the characteristics of the array being controlled. The components are: a string giving the type of array, Two integers giving the size in rows and columns and an integer giving the number outputs on the array. Other elements may be needed for certain arrays</td>
</tr>
<tr>
<td>type</td>
<td></td>
</tr>
<tr>
<td>rows</td>
<td></td>
</tr>
<tr>
<td>columns</td>
<td></td>
</tr>
<tr>
<td>outputsPerArray</td>
<td></td>
</tr>
<tr>
<td>outputArrangement</td>
<td>A structure that outlines how the array outputs are read. This includes a queue descriptor that tells where to place the pixels for processing and information on the structure of the pixel data block transferred. The block of data is described as chunks of contiguous pixels separated by a intervening pixels from other blocks. The block is described by a number of integers giving the starting row and column of the block, a row and column stride (the number of pixels to skip when storing chunks) the row and column chunk size and the total size of the final block of data in rows and columns.</td>
</tr>
<tr>
<td>picQueue</td>
<td></td>
</tr>
<tr>
<td>baseR, baseC</td>
<td></td>
</tr>
<tr>
<td>strideR, strideC</td>
<td></td>
</tr>
<tr>
<td>chunkR, chunkC</td>
<td></td>
</tr>
<tr>
<td>sizeR, sizeC</td>
<td></td>
</tr>
<tr>
<td>spcDescriptor</td>
<td>A structure that describes the configuration of the signal processor chains in the system. The components of the structure are floating point arrays that describe the gain, settling time, and offset of the signal processing chain. Included is a noise figure of merit (TBD) that will allow the quietest set of chains to be chosen when that is important.</td>
</tr>
<tr>
<td>gain</td>
<td></td>
</tr>
<tr>
<td>settingTime</td>
<td></td>
</tr>
<tr>
<td>offset</td>
<td></td>
</tr>
<tr>
<td>noiseFoM</td>
<td></td>
</tr>
<tr>
<td>waveforms</td>
<td>A descriptor for the timing waveforms to be run when running the array. These will be an array of bytes which will either describe or define the timing of the array readout. It is expected that each system will have an idiosyncratic way of describing these waveforms.</td>
</tr>
<tr>
<td>DacValueN – Float</td>
<td>An array of floating point voltage values which are to be loaded into any DAC settable voltages used to control the array. Each system will likely have a unique set of these voltages and a mapping from voltage name to DAC number should be provided in the GPX.</td>
</tr>
<tr>
<td>Min Integration Time</td>
<td>A floating point number giving the minimum integration time achievable by the system.</td>
</tr>
<tr>
<td>Base Readout Time</td>
<td>A floating point number giving the fastest possible readout time for the entire array.</td>
</tr>
<tr>
<td>spdRoiDescriptor</td>
<td>A structure that describes a “speed up” region of interest (ROI). This is provided so a system with a large array can describe a sub-array that will be read out to provide faster readout and shorter integration times. (Mostly used for IR systems without an internal cold shutter) The ROI is described by four integers giving the first row and column to be read and the size of the ROI in rows and columns.</td>
</tr>
<tr>
<td>Row0</td>
<td></td>
</tr>
<tr>
<td>Col0</td>
<td></td>
</tr>
<tr>
<td>rowSize</td>
<td></td>
</tr>
<tr>
<td>ColSize</td>
<td></td>
</tr>
<tr>
<td>binning</td>
<td>An integer value giving the binning factor for the array readout. This may be two values if the row and column binning factors are different.</td>
</tr>
</tbody>
</table>
Table 1 – Attributes (Cont.)

<table>
<thead>
<tr>
<th>Keyword Name</th>
<th>Usage/Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>intTimeSecs</td>
<td>A floating point number giving the desired integration time to use in seconds.</td>
</tr>
<tr>
<td>digAvgs</td>
<td>An integer giving the Number of Digital Averages to use while reading out the pixels</td>
</tr>
<tr>
<td>numPics</td>
<td>An integer giving the number of pictures to generate for each gpxStartExp</td>
</tr>
<tr>
<td>ROI descriptors</td>
<td>A list of structures defining the regions of interest (ROI) to read out and archive. The components of the structure are four values representing the first row and column included in the ROI and the row and column size of the ROI</td>
</tr>
<tr>
<td>rowSize</td>
<td></td>
</tr>
<tr>
<td>ColSize</td>
<td></td>
</tr>
<tr>
<td>Outputs to Read</td>
<td>An integer giving the number of outputs on the array that will be used during the readout.</td>
</tr>
<tr>
<td>PreFlash</td>
<td>A boolean value determining if the exposure sequence will include a pre-flash step.</td>
</tr>
<tr>
<td>waveFormsToRun</td>
<td>A list of the waveforms to run during this array readout.</td>
</tr>
<tr>
<td>shutterState</td>
<td>A boolean value determining if the shutter is to be opened during the integration time.</td>
</tr>
<tr>
<td>arrayPowerState</td>
<td>A boolean value determining if the array will be activated/powered-up during the exposure</td>
</tr>
<tr>
<td>intraPixelDelay</td>
<td>A floating point number giving the amount of time to allow for settling while reading each pixel</td>
</tr>
<tr>
<td>idleProcess</td>
<td>An integer tag describing how the array will be run during any idle time in the observing run</td>
</tr>
<tr>
<td>Data Disposition</td>
<td>struct disposition – procedure name Arguments - filename, directory image format, data type, data stream/queue</td>
</tr>
<tr>
<td>Pre-Processing Algorithm</td>
<td></td>
</tr>
<tr>
<td>Unscrambling Algorithm</td>
<td></td>
</tr>
<tr>
<td>Image Data Set ID</td>
<td>coAdd – integer</td>
</tr>
<tr>
<td>fSamples</td>
<td>binning - integer</td>
</tr>
<tr>
<td>intTimeSecs</td>
<td>digAvgs – integer</td>
</tr>
<tr>
<td>numPics</td>
<td>shutterState</td>
</tr>
<tr>
<td>Keyword Name</td>
<td>Usage/Explanation</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>arrayPowerState</td>
<td></td>
</tr>
<tr>
<td>Data Disposition</td>
<td></td>
</tr>
<tr>
<td>Pre-Processing Algorithm</td>
<td></td>
</tr>
<tr>
<td>Unscrambling Algorithm</td>
<td></td>
</tr>
<tr>
<td>Image Data Set ID</td>
<td></td>
</tr>
<tr>
<td>coAdds – integer</td>
<td></td>
</tr>
<tr>
<td>fSamples – integer</td>
<td></td>
</tr>
<tr>
<td>TriggerSource-</td>
<td></td>
</tr>
<tr>
<td>TriggerTimeOut</td>
<td></td>
</tr>
<tr>
<td>Image Data Set ID</td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
<tr>
<td>intTimeSecs – float</td>
<td></td>
</tr>
<tr>
<td>intTimeSecs – float</td>
<td></td>
</tr>
<tr>
<td>current Shutter State</td>
<td></td>
</tr>
<tr>
<td>Row or Y shift</td>
<td></td>
</tr>
<tr>
<td>Column or X shift</td>
<td></td>
</tr>
<tr>
<td>Units to Simulate</td>
<td></td>
</tr>
<tr>
<td>Units to Test</td>
<td></td>
</tr>
<tr>
<td>System Power State</td>
<td></td>
</tr>
<tr>
<td>System Reset Level</td>
<td></td>
</tr>
</tbody>
</table>