

DCP Data Service (DDS) Protocol Specification

Protocol Version 14

2/29/2016

Prepared For

**National Oceanic and Atmospheric Administration (NOAA)
National Environmental Satellite Data Information Service (NESDIS)**

by

Cove Software, LLC
6030 Marshalee Drive, Ste 607
Elkridge, MD 21075-5935, USA
(410) 715-1117
info@covesw.com



Table of Contents

1	INTRODUCTION	3
1.1	HISTORY OF DDS.....	4
1.2	RFC 2119 CONFORMANCE	6
1.3	BNF NOTATION	6
2	DDS PROTOCOL MESSAGES.....	8
2.1	DDS REQUEST/RESPONSE HEADERS	8
2.2	NORMAL AND ERROR RESPONSES	8
3	CONNECTING AND DISCONNECTING	11
3.1	TCP SOCKETS.....	11
3.2	AUTHENTICATION BY ASSERTION.....	12
3.3	AUTHENTICATED CONNECTION	13
3.4	DISCONNECTING	14
4	TRANSFERRING SEARCH CRITERIA TO THE SERVER	15
4.1	SEARCH CRITERIA FILE FORMAT	15
5	TRANSFERRING NETWORK LISTS TO/FROM THE SERVER.....	16
5.1	SENDING A TRANSIENT NETWORK LIST TO THE SERVER	16
5.2	RETRIEVING NETWORK LISTS FROM THE SERVER	17
5.3	NETWORK LIST FILE FORMAT.....	17
6	RETRIEVING DATA	18
6.1	RETRIEVING A SINGLE MESSAGE PER REQUEST	19
6.1.1	<i>Semantics for Until Time and Real-Time Retrieval</i>	<i>19</i>
6.2	RETRIEVING MULTIPLE MESSAGES PER REQUEST	20
6.3	EXTENDED MULTIPLE MESSAGE REQUESTS.....	20
7	STATUS AND EVENTS	22
7.1	GET EVENTS	22
7.2	GET STATUS.....	22
8	ADMINISTRATIVE FUNCTIONS.....	35
8.1	USER ADMINISTRATION.....	35
8.1.1	<i>List User Request</i>	<i>35</i>
8.1.2	<i>Set User Request.....</i>	<i>36</i>
8.1.3	<i>Remove User Request.....</i>	<i>36</i>
8.2	CONFIGURATION COMMANDS.....	37
8.2.1	<i>Return Configuration to Client</i>	<i>37</i>
8.2.2	<i>Install Configuration on Server.....</i>	<i>37</i>
8.3	OUTAGES	37
8.3.1	<i>Get Outages.....</i>	<i>37</i>
8.3.2	<i>AssertOutage.....</i>	<i>38</i>
9	REFERENCE IMPLEMENTATION.....	39

1 Introduction

DDS stands for “DCP Data Service”. It is a client/server protocol for efficiently transferring DCP data over a network. DDS is in wide use among agencies that use the GOES (Geosynchronous Operational Environmental Satellite) DCS (Data Collection System).

This document provides a description of DDS and its history. It also defines the client server protocol in detail.

1.1 History of DDS

DDS Protocol Version 1:

DDS was originally developed by Integral Systems, Inc., as part of the DOMSAT Receive Station product. The DOMSAT Receive Station collected satellite data and stored it in a circular file on the hard disk. Clients could connect using DDS and retrieve any subset of data, either historical or in real-time.

USGS, BLM, and other organizations coded their applications to act as DDS clients, pulling data from a DOMSAT system in real-time.

DDS Protocol Version 2:

In 2000, ILEX Engineering, Inc. (ILEX) produced a Java implementation of DDS for use in the LRGS (Local Readout Ground Station) DOMSAT receiver. This implementation enhances the original by allowing for the transfer of network list files. This is an important capability because it makes a client more independent from the server. The client can start a session by downloading all needed network lists. Before, the client would have to rely on persistent lists that were pre-loaded on the server.

DDS Protocol Version 3:

For LRGS Release 3.3, ILEX added a password-protection mechanism to DDS. This work was done under contract to the USGS. The mechanism uses a non-reversible hash of the passwords to prevent detection of passwords by monitoring network traffic. This document provides the details on the password exchange when a client establishes a connection.

DDS Protocol Version 4 and 5:

Ilex participated in an effort to build a replacement for the central DCP message processing system in Wallops, VA. This work was done for NOAA/NESDIS (National Oceanic and Atmospheric Administration, National Environmental Satellite Data Information Service). The new system, called “DAPS-II” will incorporate DDS as a primary mechanism for distributing DCP data to the user community. For DAPS-II, an enhancement has been added to DDS to improve performance, especially when used over a wide area network.

Protocol Version 5 is identical to 4. The additional version is added because initial server implementations of the ‘message-block’ feature were not reliable on some OS platforms. This feature was subsequently tuned and tested on a variety of platforms. Current clients are recommended to not use the message-block feature unless the server supports protocol version 5 or higher.

Successive versions are additive. None of the original features have been deprecated.

DDS Protocol Version 6:

In this version, secure administrative commands have been added. All of these commands are restricted to authenticated users that have been granted administrative privileges on the server. These commands include:

1. User Administration – may contain sub-command to add, delete, or modify DDS user accounts on the server.

DDS Protocol Version 8

The commands were added:

1. Get Status – The server returns its current status as a block of XML
2. Get Events – the server returns recent events as text messages
3. Return Configuration – The server sends a configuration file
4. Install Configuration – The client sends a configuration file to the server
5. Message Block Extended – Messages are returned as a block of compressed XML containing extended performance parameters.
6. Get Outages – The server returns a list of recent outages.
7. Assert Outage – The client tells the server to assert or reassert outages.

DDS Protocol Version 10

- Support added for Iridium messages, which have a different header structure from GOES. Servers must not send Iridium messages to clients with a version < 10.

DDS Protocol Version 11

- Added “single mode” as a settable param in search criteria.
- Added <LocalRecvTime> element in XML message block

DDS Protocol Version 12

- Added new SOURCE names to search criteria: GOES_SELFTIMED, GOES_RANDOM.

DDS Protocol Version 13 – Jan 30, 2016

- Addition of Set PW Function. This allows the server to evaluate password quality.

DDS Protocol Version 14 – Feb 29, 2016

- Support for SHA256 for the hashing function for authentication.

1.2 RFC 2119 Conformance

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119, which can be found at:

<http://www.ietf.org/rfc/rfc2119.txt?number=2119>

1.3 BNF Notation

This document uses BNF (Backus Naur Form) to define the syntax of messages sent between client and server. The following conventions are used:

<i>Notation</i>	<i>Meaning</i>
<code>::=</code>	Is defined as
<code>'literal'</code>	A literal string is enclosed in single quotation marks
<code>nonterminal</code>	Non-terminal symbols are not enclosed in quotation marks. It must be recursively defined elsewhere.
<code>one two</code>	Pipe symbol means 'or'. This rule means "one or two".
<code>{ rule }</code>	Curly brackets mean zero or more repetitions of rule.
<code>[optional]</code>	Rules in square brackets are optional.
<code>DIGIT</code>	Any ASCII digit 0 through 9
<code>CRLF</code>	ASCII Carriage Return followed by Line Feed
<code>SP</code>	ASCII Space Character
<code>STRING</code>	Any sequence of printable ASCII characters except CRLF. May contain space or tab characters.
<code>OCTET_STRING</code>	Any sequence of 8-bit binary octet values. This is only used for transferring DCP message data.
<code>(group of symbols)</code>	Parentheses used for grouping within rules.
<code># comment</code>	Characters after an un-quoted pound sign are comments.

Table 1-1: BNF Conventions Used in this Document.

Some requests and responses contain a time stamp. All time stamps MUST be in UTC and SHALL be formatted as follows:

```
time ::= YYDDDDHHMMSS
```

‘YY’ is the last two digits of the year.

‘DDD’ is the Julian day of the year (January 1 == day 1)

‘HHMMSS’ is the UTC hour, minute, and second of the day.

Integers are made up of at least one digit:

```
integer ::= DIGIT { DIGIT }
```

Hex numbers are represented by <hexstring>:

```
hexstring ::= hexdigit { hexdigit }  
hexdigit  :: DIGIT |  
           'a' | 'b' | 'c' | 'd' | 'e' | 'f' |  
           'A' | 'B' | 'C' | 'D' | 'E' | 'F' |
```

A “NAME” is an alphanumeric string that contains no whitespace. It must begin with a letter.

```
NAME ::= letter { letter | digit | underscore }  
underscore ::= '_'
```

A special identifier ‘empty’ is occasionally used to explicitly indicate a field that contains no data (i.e. zero length)

2 DDS Protocol Messages

This section describes the general features of the protocol that are observed by all message types.

2.1 DDS Request/Response Headers

Each request and response is composed of a 10-byte header followed by a variable length body. Protocol messages are constructed as follows:

```
DdsMessage ::= header body
header ::= sync type length
sync ::= 'FAF0'
type ::= octet # unique type codes defined for each message
length ::= DIGIT DIGIT DIGIT DIGIT DIGIT # 5-digit number
body ::= OCTET_STRING
```

First 4 bytes MUST be the ASCII characters “FAF0” (the last character is a zero).

The next byte contains the message type. Type-codes for each request are described below.

The next 5 bytes is a five-digit number, zero-filled. This specifies the exact number of bytes contained in the body to follow.

All client requests and server responses MUST be valid DdsMessages, as defined above. For each request, the server MUST send a single response.

The body portion of requests and responses varies with each message type and are described in the following sections.

2.2 Normal and Error Responses

A server MUST respond to a request with either a normal response or an error response. Exactly one response MUST be returned for each request.

The body portion of an error responses MUST be formatted as follows:

```
ErrorBody ::= '?' ServerCode ',' SystemCode ',' [ explanation ]
ServerCode ::= integer
SystemCode ::= integer
explanation ::= STRING # optional free-form ASCII string
```

The SystemCode is a Unix ‘errno’ value. This may be zero if the error was internal to the server. It will be non-zero if the problem was a system error, for example, attempting to retrieve a network list file that does not exist.

ServerCodes were originally designed for use on DOMSAT systems. Currently defined codes are shown in Table 2-1. Several of these codes were invented to support various iterations of DOMSAT receivers and may have no meaning to other server

implementations. Servers SHOULD refrain from defining new codes unless absolutely necessary.

<i>Name</i>	<i>Code</i>	<i>Description</i>
DSUCCESS	0	
DNOFLAG	1	Could not find start of message flag.
DDUMMY	2	Message found (and loaded) but it's a dummy.
DLONGLIST	3	Network list was too long to upload.
DARCERROR	4	Error reading archive file.
DNOCONFIG	5	Cannot attach to configuration shared memory
DNOSRCHSHM	6	Cannot attach to search shared memory
DNODIRLOCK	7	Could not get ID of directory lock semaphore
DNODIRFILE	8	Could not open message directory file
DNOMSGFILE	9	Could not open message storage file
DDIRSEMERR	10	Error on directory lock semaphore
DMSGTIMEOUT	11	Timeout waiting for new messages
DNONETLIST	12	Could not open network list file
DNOSRCHCRIT	13	Could not open search criteria file
DBADSINCE	14	Bad since time in search criteria file
DBADUNTIL	15	Bad until time in search criteria file
DBADNLIST	16	Bad network list in search criteria file
DBADADDR	17	Bad DCP address in search criteria file
DBADEMAIL	18	Bad electronic mail value in search criteria file
DBADRTRAN	19	Bad retransmitted value in search criteria file
DNLSTXCD	20	Number of network lists exceeded
DADDRXCD	21	Number of DCP addresses exceeded
DNOLRGLAST	22	Could not open last read access file
DWRONGMSG	23	Message doesn't correspond with directory entry
DNOMOREPROC	24	Can't attach: No more processes allowed
DBADDAPSSTAT	25	Bad DAPS status specified in search criteria.
DBADTIMEOUT	26	Bad TIMEOUT value in search crit file.
DCANTIOCTL	27	Cannot ioctl() the open serial port.
DUNTILDRS	28	Specified 'until' time reached
DBADCHANNEL	29	Bad GOES channel number specified in search crit
DCANTOPENSER	30	Can't open specified serial port.
DBADDCPNAME	31	Unrecognized DCP name in search criteria
DNONAMELIST	32	Cannot attach to name list shared memory.
DIIDXFILEIO	33	Index file I/O error
DNOSRCHSEM	34	Cannot attach to search semaphore
DUNTIL	35	Specified 'until' time reached
DJAVAIF	36	Error in Java - Native Interface
DNOTATTACHED	37	Not attached to LRGS native interface
DBADKEYWORD	38	Bad keyword
DPARSEERROR	39	Error parsing input file
DNONAMELISTSEM	40	Cannot attach to name list semaphore.
DBADINPUTFILE	41	Cannot open or read specified input file
DARCFILEIO	42	Archive file I/O error
DNOARCFILE	43	Archive file not opened
DICFIOCTL	44	Error on ICPl88 ioctl call
DICPIOERR	45	Error on ICPl88 I/O call
DINVALIDUSER	46	Invalid user name
DDDSAUTHFAILED	47	DDS Authentication Failure
DDDSINTERNAL	48	DDS Internal Error
DDDSFATAL	49	DDS Fatal internal server error
DNOSUCHSOURCE	50	Search criteria specified an invalid data source.
DALREADYATTACHED	51	This user is already connected and multiple connections by the same user have been diallowed.
DNOSUCHFILE	52	The client has requested a file that doesn't exist.
DTOOMANYDCPS	53	Search criteria is over this server's #DCP limit for a real-time stream
DBADPASSWORD	54	A new password does not meet local requirements.
DSTRONGREQUIRED	55	User attempted an authenticated connection with SHA, but SHA-256 is required.

Table 2-1: Currently Defined Error Codes

Table 2-2 contains the valid type-codes for DDS messages

<i>Type-Code</i>	<i>Name</i>	<i>Description</i>
a	IdHello	Client unauthenticated connect request. Server accept/reject response.
b	IdGoodbye	Client sends terminal handshake message before disconnect. Echoed back to client by server.
c	IdStatus	Client requests the servers status. Server response is a block of XML.
e	IdStop	Used to abort data retrievals that may take a long time to time-out. This command is essentially a NOOP. Server echoes this message as a response to the abort.
f	IdDcp	Client request for next DCP message. Server response containing error or DCP message.
g	IdCriteria	Client reads or writes search criteria on the server. The same type-code used for bidirectional transfer. See section 4 for details.
h	IdGetOutages	Client requests recent outages on the server. Server response contains the outage information in formatted text.
j	IdPutNetlist	Client uploads a network list to the server. Server accept/reject response.
k	IdGetNetlist	Client requests download of a network list from the server. Server response contains error or the network list.
l	IdAssertOutages	Client sends one or more outages as formatted text strings. Server accept/reject response.
m	IdAuthHello	Authenticated connect message containing hash of password. Server accept/reject response.
n	IdDcpBlock	Client request for next block of DCP messages. Server response containing multiple DCP messages in one DdsMessage
o	IdEvents	Client requests recent events on the server Server response contains events in formatted text.
p	IdRetConfig	Client requests a named configuration file on the server. Server response contains the file or error message.
q	IdInstConfig	Client sends configuration data to the server. Server accept/reject response.
r	IdDcpBlockExt	Client requests the next block of DCP messages in extended format. Server response contains DCP messages or error message.
u	IdUser	User administration commands: Add, Modify, Delete

Table 2-2: Type-Codes used in DDS Messages.

3 Connecting and Disconnecting

This section describes how connections are established and broken.

3.1 TCP Sockets

DDS is a simple client/server protocol running over TCP sockets. The server establishes a listening socket. The client connects to the port number for this socket. A new bidirectional socket is then established for communication between client and server.

By default, the server SHOULD listen on port 16003. Older implementations used 9999. Clients and servers SHOULD be coded so that the listening port is configurable.

After establishing the socket, the first request from the client MUST be one of the two authentication mechanisms described below. Any requests sent prior to a valid Authentication Exchange MUST generate an error response from the server.

3.2 Authentication by Assertion

This message type is supported by all protocol versions. However, Version 3 (and above) servers MAY disallow authentication-by-assertion if they only want to support authenticated clients. In this case the server MUST return an error to this request.

“Authentication by Assertion” means that the client simply asserts an identity by passing a username to the server. If the username matches a valid user on the server, the connection is accepted.

Authentication by Assertion is safe under the following conditions:

- The server maintains search criteria and network list files in a temporary session-directory.
- The server places limits on the size and number of network lists to be stored.
- The client does not make assumptions about what network lists are currently available on the server (i.e. it should always upload the list at the start of each session).

The type-code for IdHello is ‘a’. The body of the request and response MUST be as follows:

```
HelloRequest ::= username
username ::= NAME # no more than 80 chars
HelloResponse ::= AcceptResponse | ErrorBody
AcceptResponse ::= username [ SP ProtocolVersion ]
ProtocolVersion ::= integer
```

The body of the request contains the user name. Older implementations padded this name to 80 characters by adding spaces to the right. Servers MUST support this.

On success, the server MUST send an AcceptResponse, containing the username, and optionally, an integer representing the highest protocol version supported by this server. If the protocol version is not present in the response, the client SHOULD assume protocol version 1.

3.3 *Authenticated Connection*

This message type is supported by protocol version 3 and higher. Clients **MUST NOT** send this message to servers with a lower protocol version. Servers **SHOULD** be configurable as to whether they support and/or require authenticated connections.

An ‘authenticated hello message’ has a type-code of `IdAuthHello` = ‘m’. The body is as follows:

```
AuthHelloBody ::= username SP time SP AuthenticatorHash
username ::= NAME
# time is UTC date/time stamp in format YYDDDDHHMMSS
AuthenticatorHash ::= hexstring # 40 hex digits for SHA, 64 for SHA-256
# Response is as follows:
AuthHelloResp ::= AuthAcceptResp | ErrorBody
AuthAcceptResp ::= username SP time SP ProtocolVersion
ProtocolVersion ::= integer
```

Username must represent a valid user on the server. The time should be the current time in the UTC (GMT) time-zone. The server **SHOULD** check for the reasonableness of the time in order to prevent replay attacks. Clients and servers **MUST** disallow zero-length usernames or passwords.

The `AuthenticatorHash` is either a 40-character hex representation of a 20-byte SHA hash code, or (for Protocol Version 14) a 64-character hex representation of a 32-byte SHA-256 hash code. Clients and servers **MUST** construct the hash-code as follows:

1. Construct a preliminary SHA1 hash code with no time component. The hash should be constructed from:
 - username
 - password
 - username
 - password
2. The preliminary hash represents a shared-secret that is stored on the server, and supplied by the user.
3. The authenticator is another hash. It is constructed from:
 - username
 - preliminary hash
 - time-bytes: 4-byte integer representing time since Unix epoch, in big-endian order.
 - username
 - preliminary hash
 - time-bytes
4. Convert the binary authenticator hash to a hexadecimal string. Use capital letters for A-F.

For Protocol Version 14, the server MAY be configured to require the stronger SHA-256 as the hashing algorithm for the authenticator. Such a server SHOULD respond to an initial SHA connection with an error body containing the error code 55. The client MAY re-attempt the connection with SHA-256.

After a successful connection, both client and server must construct a one-time session key by following the same instructions as above, but substitute the authenticator string for 'username'. This session key will be used to encrypt sensitive data in all administrative functions.

3.4 *Disconnecting*

This capability is supported by all protocol versions.

The proper way to terminate a connection is to send the IdGoodbye (type='b') message, wait for the response, and then close the socket. Clients SHOULD be coded this way.

The IdGoodbye message has an empty (zero-length) body. Upon receiving such a message, the server MUST simply echo the request back to the client.

If a client simply closes the socket, the server will most-likely detect this and close the socket properly. Sometimes, particularly over WAN connections, the server may not detect this right away. Servers SHOULD implement a timeout mechanism such that clients that have issued no requests in N seconds can be disconnected. The value of N SHOULD be configurable on the server.

4 Transferring Search Criteria to the Server

Capabilities in this section apply to all version of DDS protocol.

A client specifies which messages it wants to retrieve by sending a “Search Criteria” file to the server. Search Criteria file format is described in section 4.1.

The client SHOULD transmit the desired search criteria to the server at least once prior to retrieving data. In other words, the client SHOULD NOT make any assumptions about what search criteria (if any) are in effect on the server.

Once sent, a search criteria stays in effect for the duration of the session, or until another search criteria is sent.

The type-code for IdCriteria is ‘g’.

```
SearchCritReq ::= FiftyBlanks CriteriaText
FiftyBlanks ::= 50*( SP )      # 50 ASCII space characters
CriteriaText ::= OCTET_STRING
# Response is either error or just the 50-blanks
SearchCritResp ::= ErrorBody | FiftyBlanks
```

The “FiftyBlanks” field used to contain a file-name that was used only by the client (the server simply echoed it). This is deprecated. Clients and servers should fill this field with exactly 50 space characters.

The CriteriaText is a variable length text buffer containing the file contents.

Lines in the file-data MUST be terminated by a single line-feed character.

Search Criteria files MUST NOT be longer than 16000 bytes.

On success, the server MUST respond with valid message of type IdCriteria (‘g’). The message body SHOULD contain the 50 space characters only. Note that older servers echoed the complete search criteria file. Clients SHOULD be coded to allow (and ignore) this.

Upon receiving a search criteria file, the server must evaluate the criteria and establish a session context.

4.1 Search Criteria File Format

A search criteria file is a text file containing a series of keyword-value pairs, one per line, separated by a colon:

```
KEYWORD: Value
```

The keyword/value pairs depend on the LRGS server version. Consult the search criteria file format section in the appropriate server manual.

5 Transferring Network Lists to/from the Server

This capability exists in DDS protocol version 2 and higher. Clients **MUST NOT** send this type of request to version 1 servers.

DDS Version 2 added a new capability for transferring network list files to and from a server. Version 1 relied on network lists being persistently stored and maintained on the server.

Servers **MAY** implement a separate mechanism for storing network lists persistently. This relieves clients from the burden of sending them at the start of each session. Persistent lists can be stored and maintained securely, and are available for reference by any client.

Clients **MAY** upload *transient* network lists to the server at any time in a session. Network lists transferred via DDS **SHOULD** be considered transient lists. Clients **SHOULD NOT** make any assumption regarding what transient network lists reside on the server. In particular, a client **SHOULD NOT** assume that a list uploaded in a previous session is still available on the server.

The only size limitation imposed on a transient network list by the protocol is that it must fit in a single protocol-message. The header uses 5 digits to represent the message body length. This means that transient network lists are limited to (99999 – 64) bytes.

5.1 Sending a Transient Network List to the Server

The type-code for IdPutNetlist is ‘j’. This command uploads a list from the client to the server.

```
PutNetlistReq ::= filename ListText
filename ::= NAME { SP } # Name left-justified in 64-char field
ListText ::= OCTET_STRING
# Response is either empty or an ErrorBody
PutNetlistResp ::= empty | ErrorBody
```

The file name field must be exactly 64 characters long. The name is left justified in the field and padded with blanks. The name field **SHOULD NOT** contain path separators ‘/’ or ‘\’. That is, it should be a simple filename.

On success, the server **MUST** respond with a DDS message with type IdPutNetlist (‘j’) with an empty message body.

Upon error, the server **MUST** respond with a message containing an ErrorBody with sufficient information for the client to diagnose the problem.

5.2 Retrieving Network Lists From the Server

Version 2 servers MUST implement this mechanism allowing clients to retrieve network lists. This applies to both transient lists and persistent lists.

The type-code for IdGetNetlist is 'k'. This command downloads a list from the server to the client.

```
GetNetlistReq ::= filename
filename ::= NAME { SP } # Name left-justified in a 64-char field
GetNetlistResp ::= ErrorBody | ( filename ListText )
```

Following the header is a 64-character field containing the network list file name, left justified. The name field SHOULD NOT contain path separators '/' or '\'. That is, it should be a simple filename.

On success, the server MUST respond with a DDS message of type IdGetNetlist ('k') containing:

- a 64 character field containing the network list file name, left justified.
- a variable-length field containing the network list file contents.

If the specified list is not available on the server a response of type IdGetNetlist with an ErrorBody MUST be returned.

5.3 Network List File Format

Network List Files are ASCII Files containing a DCP addresses, one per line. Each line MUST be terminated by a single line-feed character. The format of each line is as follows:

```
NetlistFile ::= { NetlistLine }
NetlistLine ::= DcpAddress [ ':' NAME [ SP Description ] ] EOL
Description ::= STRING
EOL ::= LF | CRLF
```

Example

```
CE3E13BC:WTSM5 Chippewa River Diversion Dam near Watson, MN
CE3E86DE:GLKM5 GULL LAKE ELEVATION near Brainerd, MN
CE456DFA:BIFM5 BIG FORK RIVER AT BIG FALLS, MN
CE45705E:GPOM5 LAKE KABETOGAMA AT GOLD PORTAGE, MN
CE457E8C:SSIM5 LAKE OF THE WOODS AT SPRING STEEL ISLAND, nr Warroad, MN
```

6 Retrieving Data

After connecting, authenticating, sending network lists, and sending search criteria, a client typically enters a loop where it continually polls for the next message that passes the criteria.

In all mechanisms described below for retrieving data, the server MUST only send DCP messages that match the client's search criteria.

All DCP Messages MUST start with the standard 37-byte DOMSAT header as defined in Table 6-1.

<i>Offset</i>	<i>Length</i>	<i>Type</i>	<i>Description</i>
0	8	hexstring	8 hex digit DCP address
8	11	time	Time formatted as YYDDDDHHMMSS in UTC.
19	1	char	Message type codes 'G' means a good message, '?' means a message of questionable quality. Other type codes indicate DAPS-generated status messages.
20	2	integer	2 digit signal strength. Signal Strength will be two ASCII digits and will be in the range of 32 to 57. Signal strength is the implied EIRP, assuming the pilot is a +47 dBm reference.
22	2	sign digit	A + or - sign followed by a single digit or the letter 'A'. Represents frequency offset in units of 50 Hz. A represents the maximum offset of 500 Hz.
24	1	char	Modulation Index, coded as follows: N Normal: ($60^{\circ} \pm 5^{\circ}$) L Low: ($\leq 50^{\circ}$) H High: ($\geq 70^{\circ}$)
25	1	char	Data Quality Indicator, coded as follows: N Normal: Error rate better than 10^{-6} F Fair: Error rate between 10^{-4} and 10^{-6} P Poor: Error rate worse than 10^{-4}
26	3	integer	3-digit GOES channel number, zero-filled.
29	1	char	GOES Spacecraft indicator (E or W)
30	2	hexstring	2 hex digits representing uplink carrier status.
32	5	integer	5-digit message length. This is the exact number of characters to follow.

Table 6-1: DOMSAT Header Contents.

6.1 Retrieving a Single Message per Request

This message type exists in all protocol versions.

To request a single DCP message, the client sends a DDS message of type IdDcp ('f') with an empty message body, and then waits for a response.

The server constructs a response message, again with type IdDcp ('f') followed by:

- A 40-character field containing a unique file-name that could be used to store this message on the client. This field is legacy from the original implementation.
- A variable length field containing the 37-byte DOMSAT Header followed by the DCP message.

6.1.1 Semantics for Until Time and Real-Time Retrieval

If the “until” time specified in the search criteria is reached, the server **MUST** respond with an error message with ServerCode DUNTIL (35).

If the search criteria contains no until time, this indicates that the client wishes to ‘hang on the line’, retrieving data in real-time as it becomes available. When the server receives an IdDcp request, **AND** no until time has been set, **AND** there are no new messages that meet the client’s criteria, **THEN** the server **MUST** respond with an error message with ServerCode = DMSGTIMEOUT (11). When the client receives this response, it **SHOULD** pause briefly and then try the request again.

6.2 Retrieving Multiple Messages per Request

This request type was added for protocol version 4. Clients **MUST NOT** send this request to servers that do not support protocol version 4.

To request multiple DCP messages per request, the client sends a DDS message of type `IdDcpBlock` ('n'). The request has an empty (zero-length) body.

The server **MUST** send a response of type `IdDcpBlock`. The body of the response will be either an `ErrorBody` or it will contain multiple DCP messages, back-to-back:

```
# Request body is empty
MultDcpReqBody ::= empty

# Response contains DCP messages back-to-back:
MultDcpRespBody ::= ErrorBody | MultMessages

MultMessages ::= DcpMessage { DcpMessage } # at least 1 message
DcpMessage ::= DOMSATHeader DcpMsgBody

# DOMSATHeader ::= 37-bytes as defined in table
DcpMsgBody ::= OCTET_STRING # Actual message bytes
```

The server will place messages into the response up to a maximum of 10,000 bytes. The server **MUST** only place complete DCP messages into the response. If the next message does not fit, the server **MUST** return the response and await the next request.

The “until time” and “real time retrieval” semantics described above for single message transfers also applies to multiple message requests.

The server **SHOULD NOT** delay more than 55 seconds before returning a response to the client. Hence the server **MAY** return shorter than the maximum-size response if its search engine is taking a long time to find messages matching the search criteria.

The client **MUST NOT** interpret a less-than-maximum-size response as a sign that the server is finished.

6.3 Extended Multiple Message Requests

This request/response type was added for protocol version 8. Clients **MUST NOT** send this request to servers that do not support protocol version 8.

To request multiple DCP messages per request, the client sends a DDS message of type `IdDcpBlockExt` ('r'). The request has an empty (zero-length) body.

The server **MUST** send a response of type `IdDcpBlockExt`. The body of the response will be either an `ErrorBody` or it will contain multiple DCP messages, back-to-back:

```
# Request body is empty
ExtMultDcpReqBody ::= empty

# Response contains DCP messages back-to-back:
ExtMultDcpRespBody ::= ErrorBody | ExtMultMessages

ExtMultMessages ::= GZIP( ExtMultMsgBlock )
```

Each 'ExtMultMsgBlock' is a block of XML with the format:

```
<MsgBlock>
  <DcpMsg flags=0xnnnn>
    <BinaryMsg>BASE64 (DOMSAT Header and msg data)</BinaryMsg>
    [<CarrierStart>YYYY/DDD HH:MM:SS.mmm</carrierStart>]
    [<CarrierStop>YYYY/DDD HH:MM:SS.mmm</carrierStop>]
    [<DomsatTime>YYYY/DDD HH:MM:SS.mmm</domsatTime>]
    [<DomsatSeq>NNNN</domsatSeq>]
    [<Baud>NNNN</baud>]
  </DcpMsg>

  ...additional DcpMsg blocks here

</MsgBlock>
```

The BASE64 encoding of the DOMSAT header and data is necessary to prevent the XML formatter and parsers from modifying white-space.

7 Status and Events

7.1 Get Events

```
GetEventsReqBody ::= empty
GetEventsResp ::= [ event ]
event ::= priority time msg
priority ::= 'INFO' | 'WARNING' | 'FAILURE' | 'FATAL'
time ::= YYYY/MM/DD-HH:MM:SS
msg ::= STRING
```

The server SHOULD keep a context-pointer into its event queue for each client. After connecting the pointer SHOULD be initialized to the current time. Each call to ‘Get Events’ MUST return the next events, if any, that have occurred.

7.2 Get Status

```
GetStatusReqBody ::= empty
GetStatusResp ::= Block of XML Information
```

The exact status information returned may vary with the server implementation. The following figure shows an example status block from version 5.9 of the LRGS

```
<LrgsStatusSnapshot hostname="drot-dcs">
  <systemStatus>Running</systemStatus>
  <isUsable>true</isUsable>
  <SystemTime>1172090100000</SystemTime>
  <MaxClients>100</MaxClients>
  <CurrentNumClients>23</CurrentNumClients>
  <majorVersion>5</majorVersion>
  <minorVersion>9</minorVersion>
  <ArchiveStatistics>
    <dirOldest>0</dirOldest>
    <dirNext>351434</dirNext>
    <dirWrap>0</dirWrap>
    <dirSize>12571940</dirSize>
    <oldestOffset>0</oldestOffset>
    <oldestMsgTime>1169423950</oldestMsgTime>
    <lastSeqNum>59856</lastSeqNum>
    <maxMessages>0</maxMessages>
    <maxBytes>0</maxBytes>
  </ArchiveStatistics>
  <Process slot="0" pid="29154">
    <name>127.0.0.1-29154</name>
    <type>DDS-CLI</type>
    <user>ilex</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>1171721839</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="1" pid="3">
    <name>192.168.101.177-3</name>
    <type>DDS-CLI</type>
    <user>wcdas</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090097</LastPollTime>
    <LastMsgTime>0</LastMsgTime>
```

```

    <staleCount>0</staleCount>
  </Process>
  <Process slot="2" pid="26244">
    <name>adsl-75-49-139-242.dsl.scrum01.sbcglobal.net-26244</name>
    <type>DDS-CLI</type>
    <user>testuser</user>
    <status>running</status>
    <LastSeqNum>610241</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>1172090094</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="3" pid="14168">
    <name>172.16.101.172-14168</name>
    <type>DDS-CLI</type>
    <user>wcdas</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090081</LastPollTime>
    <LastMsgTime>1170151727</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="4" pid="36065">
    <name>mvr fw.mvr.usace.army.mil-36065</name>
    <type>DDS-CLI</type>
    <user>cemvr</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>1169710416</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="5" pid="12">
    <name>pro-lrqs2.pactide.noaa.gov-12</name>
    <type>DDS-CLI</type>
    <user>testuser</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>0</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="6" pid="26279">
    <name>ecyhqea01.ecy.wa.gov-26279</name>
    <type>DDS-CLI</type>
    <user>testuser</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>1170348631</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="7" pid="30091">
    <name>exeuntcha.tva.gov-30091</name>
    <type>DDS-CLI</type>
    <user>tva</user>
    <status>running</status>
    <LastSeqNum>0</LastSeqNum>
    <LastPollTime>1172090077</LastPollTime>
    <LastMsgTime>1171982931</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>
  <Process slot="8" pid="37159">
    <name>gnae911.nae.usace.army.mil-37159</name>
    <type>DDS-CLI</type>
    <user>testuser</user>
    <status>running</status>
    <LastSeqNum>39207</LastSeqNum>
    <LastPollTime>1172090099</LastPollTime>
    <LastMsgTime>1172090092</LastMsgTime>
    <staleCount>0</staleCount>
  </Process>

```

```

</Process>
<Process slot="9" pid="32524">
  <name>166-70-175-100.ip.xmission.com-32524</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090099</LastPollTime>
  <LastMsgTime>1169710416</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="10" pid="29155">
  <name>127.0.0.1-29155</name>
  <type>DDS-CLI</type>
  <user>ilex</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090099</LastPollTime>
  <LastMsgTime>1171988239</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="11" pid="38100">
  <name>c-69-250-74-32.hsd1.md.comcast.net-38100</name>
  <type>DDS-CLI</type>
  <user>ilex</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090100</LastPollTime>
  <LastMsgTime>1169710416</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="12" pid="26315">
  <name>159.87.115.4-26315</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090099</LastPollTime>
  <LastMsgTime>1171960891</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="13" pid="21">
  <name>hadsdms2.nws.noaa.gov-21</name>
  <type>DDS-CLI</type>
  <user>nwsohd</user>
  <status>running</status>
  <LastSeqNum>1925146</LastSeqNum>
  <LastPollTime>1172090087</LastPollTime>
  <LastMsgTime>1172090092</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="14" pid="36272">
  <name>mvr fw.mvr.usace.army.mil-36272</name>
  <type>DDS-CLI</type>
  <user>cemvr</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090097</LastPollTime>
  <LastMsgTime>1169710416</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="21" pid="26249">
  <name>www.ilexengineering.com-26249</name>
  <type>DDS-CLI</type>
  <user>ilex</user>
  <status>running</status>
  <LastSeqNum>610241</LastSeqNum>
  <LastPollTime>1172090099</LastPollTime>
  <LastMsgTime>1172090092</LastMsgTime>
  <staleCount>0</staleCount>
</Process>

```



```

<Process slot="23" pid="26250">
  <name>www.ilexengineering.com-26250</name>
  <type>DDS-CLI</type>
  <user>ilex</user>
  <status>running</status>
  <LastSeqNum>0</LastSeqNum>
  <LastPollTime>1172090094</LastPollTime>
  <LastMsgTime>1170348631</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="24" pid="26248">
  <name>216.9.77.210-26248</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>610257</LastSeqNum>
  <LastPollTime>1172089801</LastPollTime>
  <LastMsgTime>1172090094</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="25" pid="26256">
  <name>165.127.23.2-26256</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>610262</LastSeqNum>
  <LastPollTime>1172088012</LastPollTime>
  <LastMsgTime>1172090092</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="26" pid="26264">
  <name>142.94.10.239-26264</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>460873</LastSeqNum>
  <LastPollTime>1172088011</LastPollTime>
  <LastMsgTime>1172090094</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="27" pid="26258">
  <name>165.127.23.2-26258</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>610279</LastSeqNum>
  <LastPollTime>1172088012</LastPollTime>
  <LastMsgTime>1172090094</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="28" pid="26260">
  <name>208.187.174.43-26260</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>610253</LastSeqNum>
  <LastPollTime>1172088011</LastPollTime>
  <LastMsgTime>1172090094</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<Process slot="29" pid="26263">
  <name>208.187.174.42-26263</name>
  <type>DDS-CLI</type>
  <user>testuser</user>
  <status>running</status>
  <LastSeqNum>610257</LastSeqNum>
  <LastPollTime>1172088003</LastPollTime>
  <LastMsgTime>1172090094</LastMsgTime>
  <staleCount>0</staleCount>
</Process>
<MaxDownlinks>32</MaxDownlinks>

```

```

<DownLink slot="0" name="DomsatRecv">
  <type>1</type>
  <StatusCode>2</StatusCode>
  <status>Active</status>
  <LastMsgRecvTime>1172090099</LastMsgRecvTime>
  <LastSeqNum>59856</LastSeqNum>
  <BER>89</BER>
  <Quality hour="0">
    <numGood>14275</numGood>
    <numDropped>102</numDropped>
  </Quality>
  <Quality hour="1">
    <numGood>14061</numGood>
    <numDropped>88</numDropped>
  </Quality>
  <Quality hour="2">
    <numGood>13872</numGood>
    <numDropped>99</numDropped>
  </Quality>
  <Quality hour="3">
    <numGood>14226</numGood>
    <numDropped>82</numDropped>
  </Quality>
  <Quality hour="4">
    <numGood>14128</numGood>
    <numDropped>93</numDropped>
  </Quality>
  <Quality hour="5">
    <numGood>14004</numGood>
    <numDropped>82</numDropped>
  </Quality>
  <Quality hour="6">
    <numGood>14150</numGood>
    <numDropped>102</numDropped>
  </Quality>
  <Quality hour="7">
    <numGood>13979</numGood>
    <numDropped>107</numDropped>
  </Quality>
  <Quality hour="8">
    <numGood>13933</numGood>
    <numDropped>111</numDropped>
  </Quality>
  <Quality hour="9">
    <numGood>14309</numGood>
    <numDropped>93</numDropped>
  </Quality>
  <Quality hour="10">
    <numGood>13989</numGood>
    <numDropped>84</numDropped>
  </Quality>
  <Quality hour="11">
    <numGood>13883</numGood>
    <numDropped>93</numDropped>
  </Quality>
  <Quality hour="12">
    <numGood>14258</numGood>
    <numDropped>97</numDropped>
  </Quality>
  <Quality hour="13">
    <numGood>14038</numGood>
    <numDropped>91</numDropped>
  </Quality>
  <Quality hour="14">
    <numGood>13924</numGood>
    <numDropped>101</numDropped>
  </Quality>
  <Quality hour="15">
    <numGood>14248</numGood>
    <numDropped>115</numDropped>
  </Quality>

```

```

<Quality hour="16">
  <numGood>14237</numGood>
  <numDropped>81</numDropped>
</Quality>
<Quality hour="17">
  <numGood>14072</numGood>
  <numDropped>95</numDropped>
</Quality>
<Quality hour="18">
  <numGood>14257</numGood>
  <numDropped>104</numDropped>
</Quality>
<Quality hour="19">
  <numGood>14120</numGood>
  <numDropped>92</numDropped>
</Quality>
<Quality hour="20">
  <numGood>8491</numGood>
  <numDropped>65</numDropped>
</Quality>
<Quality hour="21">
  <numGood>14320</numGood>
  <numDropped>99</numDropped>
</Quality>
<Quality hour="22">
  <numGood>14034</numGood>
  <numDropped>109</numDropped>
</Quality>
<Quality hour="23">
  <numGood>13979</numGood>
  <numDropped>102</numDropped>
</Quality>
</DownLink>
<DownLink slot="1" name="DDS-Recv:Main">
  <type>5</type>
  <StatusCode>2</StatusCode>
  <status>Active</status>
  <LastMsgRecvTime>1172090099</LastMsgRecvTime>
  <Quality hour="0">
    <numGood>14253</numGood>
    <numDropped>160</numDropped>
  </Quality>
  <Quality hour="1">
    <numGood>14041</numGood>
    <numDropped>148</numDropped>
  </Quality>
  <Quality hour="2">
    <numGood>13838</numGood>
    <numDropped>163</numDropped>
  </Quality>
  <Quality hour="3">
    <numGood>14197</numGood>
    <numDropped>143</numDropped>
  </Quality>
  <Quality hour="4">
    <numGood>14107</numGood>
    <numDropped>138</numDropped>
  </Quality>
  <Quality hour="5">
    <numGood>13983</numGood>
    <numDropped>133</numDropped>
  </Quality>
  <Quality hour="6">
    <numGood>14122</numGood>
    <numDropped>156</numDropped>
  </Quality>
  <Quality hour="7">
    <numGood>13964</numGood>
    <numDropped>154</numDropped>
  </Quality>
  <Quality hour="8">

```

```

        <numGood>13920</numGood>
        <numDropped>162</numDropped>
    </Quality>
    <Quality hour="9">
        <numGood>14277</numGood>
        <numDropped>150</numDropped>
    </Quality>
    <Quality hour="10">
        <numGood>13971</numGood>
        <numDropped>125</numDropped>
    </Quality>
    <Quality hour="11">
        <numGood>13854</numGood>
        <numDropped>140</numDropped>
    </Quality>
    <Quality hour="12">
        <numGood>14234</numGood>
        <numDropped>156</numDropped>
    </Quality>
    <Quality hour="13">
        <numGood>14013</numGood>
        <numDropped>146</numDropped>
    </Quality>
    <Quality hour="14">
        <numGood>13876</numGood>
        <numDropped>147</numDropped>
    </Quality>
    <Quality hour="15">
        <numGood>14230</numGood>
        <numDropped>170</numDropped>
    </Quality>
    <Quality hour="16">
        <numGood>14225</numGood>
        <numDropped>136</numDropped>
    </Quality>
    <Quality hour="17">
        <numGood>14052</numGood>
        <numDropped>155</numDropped>
    </Quality>
    <Quality hour="18">
        <numGood>14248</numGood>
        <numDropped>169</numDropped>
    </Quality>
    <Quality hour="19">
        <numGood>14106</numGood>
        <numDropped>149</numDropped>
    </Quality>
    <Quality hour="20">
        <numGood>8485</numGood>
        <numDropped>92</numDropped>
    </Quality>
    <Quality hour="21">
        <numGood>14301</numGood>
        <numDropped>152</numDropped>
    </Quality>
    <Quality hour="22">
        <numGood>14017</numGood>
        <numDropped>163</numDropped>
    </Quality>
    <Quality hour="23">
        <numGood>13951</numGood>
        <numDropped>151</numDropped>
    </Quality>
</DownLink>
<DownLink slot="2" name="DRGS-Recv:Main">
    <type>4</type>
    <StatusCode>2</StatusCode>
    <status>Active</status>
    <LastMsgRecvTime>0</LastMsgRecvTime>
    <Quality hour="0">
        <numGood>0</numGood>

```

```
</Quality>
<Quality hour="1">
  <numGood>0</numGood>
</Quality>
<Quality hour="2">
  <numGood>0</numGood>
</Quality>
<Quality hour="3">
  <numGood>0</numGood>
</Quality>
<Quality hour="4">
  <numGood>0</numGood>
</Quality>
<Quality hour="5">
  <numGood>0</numGood>
</Quality>
<Quality hour="6">
  <numGood>0</numGood>
</Quality>
<Quality hour="7">
  <numGood>0</numGood>
</Quality>
<Quality hour="8">
  <numGood>0</numGood>
</Quality>
<Quality hour="9">
  <numGood>0</numGood>
</Quality>
<Quality hour="10">
  <numGood>0</numGood>
</Quality>
<Quality hour="11">
  <numGood>0</numGood>
</Quality>
<Quality hour="12">
  <numGood>0</numGood>
</Quality>
<Quality hour="13">
  <numGood>0</numGood>
</Quality>
<Quality hour="14">
  <numGood>0</numGood>
</Quality>
<Quality hour="15">
  <numGood>0</numGood>
</Quality>
<Quality hour="16">
  <numGood>0</numGood>
</Quality>
<Quality hour="17">
  <numGood>0</numGood>
</Quality>
<Quality hour="18">
  <numGood>0</numGood>
</Quality>
<Quality hour="19">
  <numGood>0</numGood>
</Quality>
<Quality hour="20">
  <numGood>0</numGood>
</Quality>
<Quality hour="21">
  <numGood>0</numGood>
</Quality>
<Quality hour="22">
  <numGood>0</numGood>
</Quality>
<Quality hour="23">
  <numGood>0</numGood>
</Quality>
</DownLink>
```

```

<DownLink slot="3" name="DDS:CDADATA">
  <type>8</type>
  <StatusCode>5</StatusCode>
  <status>Real-Time</status>
  <LastMsgRecvTime>1172090099</LastMsgRecvTime>
  <Quality hour="0">
    <numGood>14253</numGood>
    <numDropped>160</numDropped>
  </Quality>
  <Quality hour="1">
    <numGood>14041</numGood>
    <numDropped>148</numDropped>
  </Quality>
  <Quality hour="2">
    <numGood>13838</numGood>
    <numDropped>163</numDropped>
  </Quality>
  <Quality hour="3">
    <numGood>14197</numGood>
    <numDropped>143</numDropped>
  </Quality>
  <Quality hour="4">
    <numGood>14107</numGood>
    <numDropped>138</numDropped>
  </Quality>
  <Quality hour="5">
    <numGood>13983</numGood>
    <numDropped>133</numDropped>
  </Quality>
  <Quality hour="6">
    <numGood>14122</numGood>
    <numDropped>156</numDropped>
  </Quality>
  <Quality hour="7">
    <numGood>13964</numGood>
    <numDropped>154</numDropped>
  </Quality>
  <Quality hour="8">
    <numGood>13920</numGood>
    <numDropped>162</numDropped>
  </Quality>
  <Quality hour="9">
    <numGood>14277</numGood>
    <numDropped>150</numDropped>
  </Quality>
  <Quality hour="10">
    <numGood>13971</numGood>
    <numDropped>125</numDropped>
  </Quality>
  <Quality hour="11">
    <numGood>13854</numGood>
    <numDropped>140</numDropped>
  </Quality>
  <Quality hour="12">
    <numGood>14234</numGood>
    <numDropped>156</numDropped>
  </Quality>
  <Quality hour="13">
    <numGood>14013</numGood>
    <numDropped>146</numDropped>
  </Quality>
  <Quality hour="14">
    <numGood>13876</numGood>
    <numDropped>147</numDropped>
  </Quality>
  <Quality hour="15">
    <numGood>14230</numGood>
    <numDropped>170</numDropped>
  </Quality>
  <Quality hour="16">
    <numGood>14225</numGood>

```

```

    <numDropped>136</numDropped>
  </Quality>
<Quality hour="17">
  <numGood>14052</numGood>
  <numDropped>155</numDropped>
</Quality>
<Quality hour="18">
  <numGood>14248</numGood>
  <numDropped>169</numDropped>
</Quality>
<Quality hour="19">
  <numGood>14106</numGood>
  <numDropped>149</numDropped>
</Quality>
<Quality hour="20">
  <numGood>8485</numGood>
  <numDropped>92</numDropped>
</Quality>
<Quality hour="21">
  <numGood>14301</numGood>
  <numDropped>152</numDropped>
</Quality>
<Quality hour="22">
  <numGood>14017</numGood>
  <numDropped>163</numDropped>
</Quality>
<Quality hour="23">
  <numGood>13951</numGood>
  <numDropped>151</numDropped>
</Quality>
</DownLink>
<DownLink slot="4" name="DDS:CDABACKUP">
  <type>8</type>
  <StatusCode>5</StatusCode>
  <status>Ready</status>
  <LastMsgRecvTime>1171682030</LastMsgRecvTime>
  <Quality hour="0">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="1">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="2">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="3">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="4">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="5">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="6">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="7">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="8">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="9">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="10">
    <numGood>0</numGood>
  </Quality>
  <Quality hour="11">
    <numGood>0</numGood>
  </Quality>

```

```

</Quality>
<Quality hour="12">
  <numGood>0</numGood>
</Quality>
<Quality hour="13">
  <numGood>0</numGood>
</Quality>
<Quality hour="14">
  <numGood>0</numGood>
</Quality>
<Quality hour="15">
  <numGood>0</numGood>
</Quality>
<Quality hour="16">
  <numGood>0</numGood>
</Quality>
<Quality hour="17">
  <numGood>0</numGood>
</Quality>
<Quality hour="18">
  <numGood>0</numGood>
</Quality>
<Quality hour="19">
  <numGood>0</numGood>
</Quality>
<Quality hour="20">
  <numGood>0</numGood>
</Quality>
<Quality hour="21">
  <numGood>0</numGood>
</Quality>
<Quality hour="22">
  <numGood>0</numGood>
</Quality>
<Quality hour="23">
  <numGood>0</numGood>
</Quality>
</DownLink>
<Quality hour="0">
  <numGood>14277</numGood>
  <numDropped>136</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="1">
  <numGood>14067</numGood>
  <numDropped>123</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="2">
  <numGood>13873</numGood>
  <numDropped>136</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="3">
  <numGood>14225</numGood>
  <numDropped>122</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="4">
  <numGood>14130</numGood>
  <numDropped>122</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="5">
  <numGood>14011</numGood>
  <numDropped>110</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="6">
  <numGood>14150</numGood>
  <numDropped>136</numDropped>

```



```

    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="7">
    <numGood>13987</numGood>
    <numDropped>138</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="8">
    <numGood>13941</numGood>
    <numDropped>146</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="9">
    <numGood>14307</numGood>
    <numDropped>125</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="10">
    <numGood>13994</numGood>
    <numDropped>110</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="11">
    <numGood>13881</numGood>
    <numDropped>119</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="12">
    <numGood>14260</numGood>
    <numDropped>135</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="13">
    <numGood>14037</numGood>
    <numDropped>127</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="14">
    <numGood>13899</numGood>
    <numDropped>127</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="15">
    <numGood>14253</numGood>
    <numDropped>150</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="16">
    <numGood>14245</numGood>
    <numDropped>116</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="17">
    <numGood>14075</numGood>
    <numDropped>133</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="18">
    <numGood>14266</numGood>
    <numDropped>150</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="19">
    <numGood>14129</numGood>
    <numDropped>127</numDropped>
    <numRecovered>0</numRecovered>
  </Quality>
  <Quality hour="20">
    <numGood>8494</numGood>
    <numDropped>85</numDropped>
    <numRecovered>0</numRecovered>

```

```
</Quality>
<Quality hour="21">
  <numGood>14320</numGood>
  <numDropped>132</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="22">
  <numGood>14038</numGood>
  <numDropped>142</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<Quality hour="23">
  <numGood>13980</numGood>
  <numDropped>126</numDropped>
  <numRecovered>0</numRecovered>
</Quality>
<domsatDropped>
  1 0 0 2 14 4123 0 2165 13 0 0 2 8341 1855 1 0 1 0 1 1 0 6 1 1
</domsatDropped>
</LrgsStatusSnapshot>
```

8 Administrative Functions

Administrative functions SHALL only be granted to DDS users who have successfully authenticated to the server as described in section 3.3. Furthermore the server MUST restrict these commands to users who have been granted a special administrative privilege.

8.1 User Administration

A ‘user administration request’ has a type-code of `IdUser = ‘u’`. The body is as follows:

```
userRequestBody ::= listRequest | setRequest | removeRequest
listRequest ::= 'list'
setRequest ::= 'set' SP username SP encryptAuth SP roles SP props
username ::= NAME # no more than 80 chars
encryptAuth ::= '-' | NAME # Encrypted pw or '-' to leave unchanged
roles ::= NAME [',' roles] # One or more roles for this user
props ::= NAME '=' NAME [',' props] # 0 or more name=value pairs
removeRequest ::= 'rm' NAME
```

For Protocol Version 13, the following modifications have been made:

```
userRequestBody ::= listRequest | setRequest | removeRequest | pwRequest
pwRequest ::= username SP encryptPassword
encryptPassword ::= NAME # password encoded as described below
```

A user may set his/her own password. And administrator may set anyone’s password.

This means that there are two ways for a remote user (or administrator) to change the authenticator: The legacy `setRequest` containing `encryptAuth`, or the version-13 `pwRequest` containing the encrypted password. When the `pwRequest` type is used, the server must build the authenticator for storage as describe in section 3.3.

The server MAY be configured to allow only the `pwRequest` method. This method allows the server to verify that the password meets length and complexity requirements.

The `encryptedPassword` field is a BASE64 Encoding of the password encrypted with the session key. Zero length passwords are invalid and MUST be rejected by the server.

8.1.1 List User Request

As described above, the client requests that the server list all of its users. The server response is as follows:

```
listResponse ::= userLine [ listResponse]
userLine ::= username SP pwIndicator SP roles SP props
pwIndicator ::= '+' | '-' # + means password present, - means not
```

8.1.2 Set User Request

The set user request is used to create new users or modify existing ones.

The 'encryptAuth' field is either '-', meaning to leave the authenticator unchanged for this user; or it is a DES-Encrypted, base-64 encoded authenticator to be used by this user for future connections. DES-Encryption and decryption is done using the one-time session key that results from an authenticated connection.

8.1.3 Remove User Request

```
listResponse ::= ErrorBody | STRING
```

The response is simply a string stating that the user was removed.

8.2 Configuration Commands

8.2.1 Return Configuration to Client

```
returnCfgRequestBody ::= cfgfiletype
cfgfiletype ::= 'lrgs' | 'ddsrecv' | 'drgs' | 'netlist-list' |
               'netlist:' + filename
returnCfgResponseBody ::= filedata
filedata ::= ErrorBody | OCTETSTRING
```

The client requests a particular configuration from the server. The ‘cfgfiletype’ definition shows the currently supported file-types. The responses is an error message or the file contents.

The server **MUST** restrict this command to authenticated users who have been granted administrative privilege.

The network lists returned in this command are the *shared* network lists available to all users.

8.2.2 Install Configuration on Server

```
installCfgRequestBody ::= cfgfiletype + (pad to 64-bytes)
installCfgResponseBody ::= ErrorBody | OCTETSTRING
```

The responses is an error message or the file contents.

The server **MUST** restrict this command to authenticated users who have been granted administrative privilege.

The network lists installed by this command are *shared* network lists available to all users.

8.3 Outages

An ‘outage’ is a data-loss event on the server. These are triggered by various conditions on the server. The client may retrieve currently known outages or may assert new outages.

```
getOutageRequest ::= [ starttime [ endtime ] ]
starttime ::= YYYY/DDD-HH:MM:SS
endtime ::= YYYY/DDD-HH:MM:SS
getOutageResponse ::= ErrorBody | GZIP( OutageXmlData )
```

8.3.1 Get Outages

The server **MUST** restrict this command to authenticated users who have been granted administrative privilege.

Outages are returned as a Gzipped block of XML data in the following format:

```
<outageList>
  <outage outageId="nnnn" outageType="typestr">
```

```
    <beginTime>YYYY/DDD HH:MM:SS</beginTime>
  [ <endTime>YYYY/DDD HH:MM:SS</endTime> ]
  [ <sourceId>nnnn</sourceId> ]
  [ <sourceName>name</sourceName> ]
  [ <dcAddress>addr</dcAddress> ]
  [ <beginSeq>nnnn</beginSeq> ]
  [ <endSeq>nnnn</endSeq> ]
</outage>
</outageList>
```

Different types of outages may have different entities in its data. The above shows all of the possible outage data.

8.3.2 AssertOutage

The server **MUST** restrict this command to authenticated users who have been granted administrative privilege.

The body of this message is a GZipped block of XML as shown above.

9 Reference Implementation

A reference implementation of DDS is included in the LRGS (Local Readout Ground Station) code, developed by Ilex Engineering, Inc. The client software is 100% Java. The server software contains some native code and is written to run on a LRGS/DOMSAT receiver.

The LRGS software was written under contract to the NOAA NESDIS, USGS, USACE.