Web Network

Distribution Protocol

Specification (WNDP)

Version 5.00.01

August 2019

Table of Contents

Overview

Introduction

This document provides a description of the Web Network Distribution Protocol (WNDP) in the form of a client-side tutorial. The Protocol Specification is defined in Appendix A.

Throughout the document are sections marked as “design notes”. These are not part of the protocol specification. They are discussion of issues that were considered and why they were resolved in the manner documented.

Background Information

Note that the standard C notation for hex numbers is used throughout (e.g., 0xA0 = 160 decimal).

The following format will be used to illustrate the data being exchanged from server to client:

```
00000000 00000000 00000000 00000000 { ……………. }
```

where each 4-byte quantity will be space-separated for a total of 16 bytes per line. Each line will be followed by the ASCII representation of each of the 16 bytes, where ‘.’ will be used for non-printable ASCII characters.

Functional Description

Introduction

WNDP is used to transmit information between the WNDP server (WNDPSVR) program and an attaching application (client). Its primary purpose is to allow the transfer of XMLnews items to the client application. It also allows the transfer of associated files (e.g., JPEG pictures, audio clips, etc.).

Applications that are to process the XML documents being provided by WNDPSVR establish a reliable connection to the WNDPSVR program. They then use WNDP to control the transfer of the XML documents (and any associated files) from the WNDPSVR program. Additionally, applications use WNDP to provide configuration information that may affect what XML documents are transferred and in what manner they are transferred.

The expectation of WNDP is that clients connect to the WNDPSVR program and ingest messages as fast as they become available. WNDP assumes that clients do not disconnect except when a temporary failure occurs on the TCP session and then, they will perform the appropriate action to re-establish the connection. It is always incumbent upon the client application to initiate a connection.

All code examples contained in the tutorial are currently written for a UNIX client application.

Design Note

The assumption is that the client and server are typically located on the same machine or on a machine on a local LAN segment. This means that security of the connection is not of great concern and that the connection is reliable and not subject to loss that even a TCP connection experiences on the Internet. So, while loss of the TCP connection is allowed for, it is assumed to be a rare occurrence. Similarly, encryption and or compression of the data are not envisioned nor is there anything done to prevent hacking or spoofing.

Also note that the protocol doesn’t make assumptions about the format of the files. It only assumes that there is some filename that can be associated with the file. It also assumes that the files have a linear structure (e.g., they are not Macintosh files).

WNDP Transport

The transport protocol for WNDP is TCP/IP. The client application initiates the TCP connection to the WNDPSVR program. This connection occurs on port 39030 by default. WNDP can accept connections on a different port, if so required. Once the connection has been established, WNDP begins.

WNDP Overview

WNDP is a client-server model. The client application program initiates requests that are fulfilled by the WNDPSVR program (the server).

Commands have the form:

XXXX [<args>]<NULL>

Where "XXXX" is a 1 to 4-character command. The command is followed (optionally) by 0 to N space separated arguments. The command is terminated by a NULL character (single byte of 0x00).

Replies consist of a response code followed by an optional message followed by a NULL character. The commands are of the form:

<response code> [<response item>] <NULL>

except in the case of BLK data (returned from the FILE command) which will not have a <NULL> terminator. This is not required since each BLK data message will be preceded with the corresponding size in bytes of the block. (Refer to "FILE" command for more information.)

The response codes are single characters as follows:

+	Success
-	Error
!	Logged in (reserved for future use)

Protocol Initiation

The first step for a WNDP client application is to open the TCP/IP socket to the WNDPSVR program. It is recommended that the WNDP server address and well-known port be loaded at run time from a configuration file. This design lends itself for minimal changes if either of these parameters needs to change in the future.

Assume WNDP_server and WNDP_port have been loaded for the following example:

```
/* The following routine is used to establish a TCP socket
connection to a WNDPSVR application.  This routine will be
referred to in later examples. */
int wndp_tcp_init ()
{
   int optval;

   /* Open TCP socket. */
   if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
       /* Log error and fail if we get here. */
       return -1;
   }

   memset (&saddr, 0, sizeof (saddr));

   saddr.sin_family = AF_INET;
   saddr.sin_port = htons (WNDP_port);
   saddr.sin_addr.s_addr = inet_addr (WNDP_server);

   /* Set TCP keep alive timer. */
   optval = 1;
   if (setsockopt (sockfd, SOL_SOCKET, SO_KEEPALIVE,
       (char *) &optval, sizeof (optval)) < 0) {
       /* Log error and fail if we get here. */
       return -1;
   }

   /* Setup socket to be non-blocking. */
   if (fcntl (sockfd, F_SETFL, FNDELAY) < 0) {
       /* Unable to setup socket as non-blocking. */
       /* Log error and fail if we get here. */
       return -1;
   }

   /* The following is an optional socket option used to
   specify a larger receive buffer size than the TCP default. */
   optval = specify new size here;
   if (setsockopt (sockfd, SOL_SOCKET, SO_RCVBUF,
           (char *) &optval, sizeof (optval)) < 0) {
       /* Log error and fail if we get here. */
       return -1;
   }

   /* Connect to the open socket at this point. */
   sts = connect (sockfd, (struct sockaddr *) &saddr,
        sizeof(saddr));
   if (sts < 0) {
       if (errno == EINPROGRESS) {
       /* If the TCP socket is set to non-blocking, connect will
       return immediately with EINPROGRESS, but the TCP three-
       way handshake will continue.  Use select to make sure
       that it has successfully completed. */
       FD_ZERO (&rset);
```

```
        FD_SET (sockfd, &rset);
        wset = rset;
        tval.tv_sec = 1;
        tval.tv_usec = 0;

        if (select (sockfd+1, &rset, &wset, NULL, &tval) == 0) {
            close (sockfd);
            errno = ETIMEDOUT;
            /* Log error and fail if we get here. */
        }
        if (FD_ISSET (sockfd, &rset) ||
            FD_ISSET (sockfd, &wset)) {
            len = sizeof (error);
            if (getsockopt (sockfd, SOL_SOCKET, SO_ERROR,
                            (char *) &error, &len) < 0) {
                /* Log error and fail if we get here. */
                return -1;
            }
        }
        else {
            /* Log error and fail if we get here. */
            return -1;
        }
    }
    return sockfd;
}
```

Once the TCP connection has been established between the WNDPSVR program and the client application (the connection is initiated by the client application), the WNDPSVR program immediately responds with one of the following:

+WAVO WNDP vY.YY.YY

- WAVO WNDP (100) Service unavailable

The first message is used if the WNDPSVR program is willing to accept the TCP connection. The "Y.YY.YY" is the version of the protocol being used. The current protocol level is 3.00.00.

The second message is used if the WNDPSVR program is unwilling to accept the TCP connection. An optional "message" may be included to indicate why the connection is being refused. The message may include CRLF's.

Design Note

Currently, the UNIX WNDP Server allows for a total of 100 simultaneous client connections. This value can be changed by the system administrator and is based on hardware constraints of the UNIX server. Upon connection of the 101st client, the above error message (100) will be returned to the client and the connection will be closed.

Application Commands

The application can send any of the following commands:

USER – Login name

PSWD – Password

CLAS – Specify which classification of data is desired

FLTR – Specify from which providers/services data is desired

FROM – Set starting position

RQST – Request next message

FILE – Request data file

DSTR – Obtain messages names with their data streams

VRSN – Specify version of the WNDP protocol that client will use

CNFG – Configuration parameters

The following commands are no longer supported (v2.00.00 and beyond):

HRTB – heartbeat

DONE – terminates connection

Each command must be capitalized. All commands are in the UTF-8 character set. These commands are described in detail in the Protocol Detail section. Any other commands will be responded to with the reply:

- UNKN (101) Bad request ‘xxxx’<NULL>

where “xxxx” is the invalid client request.

The TCP connection is not terminated when an unknown command is received.

Design Note

It is understood for the ASCII character set (values 0x00 through 0xfd), UTF-8 and ASCII are the same (single-byte) characters. UTF-8 is also the default for XML documents. Therefore, specifying UTF-8 as the character set allows use of the current ASCII code implementations “as is”. However, this doesn’t affect the WNDP. The data transferred in the block is binary. The surrounding protocol can be in UTF-8 format without impacting the actual transfer.

Error Responses

Error responses typically have a minus sign ("-") followed by a command followed by an error string. All error strings will be as follows:

(nnn) error explanation

A 3-digit number is in parentheses. This is followed by an error explanation. The possible values of the 3-digit numbers are contained in the Error Appendix of this document. Only the error numbers are significant and only the number needs to be interpreted by the software modules. The error explanation is a UTF-8 string which is for human consumption and is mostly available for debugging purposes.

Example Client Application

```
/* Try forever to establish the TCP connection with the WNDP
server.  Note that wndp_tcp_init() is shown above in the
tutorial. */

int wndp_establish_connection ()
{
    int socket;

    while ((socket = wndp_tcp_init ()) < 0) {
        /* Log error and then retry. */
        sleep (60);
    }
    return socket;
}

int main ()
{
    int socket, maxfd;
    fd_set rset;
    struct timeval timeout;

    /* Open TCP socket. */
    socket = wndp_establish_connection();
    maxfd = socket + 1;

    /* Set client timeout to 12 seconds (greater than 8)
    as shown in “RQST” Command in Appendix A. */
    timeout.tv_usec = 0;
    timeout.tv_sec = 12;

    while (1) {

        FD_ZERO (rset);
        FD_SET (socket, &rset);

        while (select (maxfd, &rset, NULL, NULL, &timeout) < 0) {
            if (errno != EINTR) {
                /* Error condition.  Try closing socket and
                reestablishing a new connection. */
                close (socket);
                socket = wndp_establish_connection();
                break;
            }
        }

        if (FD_ISSET (socket, &rset)) {
            /* Call routine here to handle incoming WNDP data. */
        }
        else {
            /* Perform timeout processing here. */
        }
    }
}
```

Appendix A (WNDP Protocol Specification)

"USER" Command

The USER command is used to establish a level of security on the connection. Different connections may desire different services. The WNDPSVR program may require the application to provide a User name and a password before certain services are allowed.

The format of the command is:

USER <user-name><NULL>

The <user-name> is assumed to be all characters following the single space character (which must follow the string "USER") up to, but not including, the <NULL> character. The <user-name> may not include a <NULL> character. The characters allowed in the <user-name> will typically match the standard for the operating system (e.g., WNDPSVR on Microsoft Windows will require the <user-name> character set to match the character set for logins under Microsoft Windows).

If the USER command is accepted, the WNDPSVR program will respond with:

+USER<NULL>

Note that if a password is required (not part of Version 2 of the protocol), then acceptance of the USER does not indicate that the client has successfully completed connection to the system.

If the WNDPSVR program does not recognize the user name, it will respond with:

-USER (200) Invalid user name '*user-name*'<NULL>

WNDPSVR requires that a client be successfully logged in before processing any other commands. The TCP connection will not be terminated, but unless a valid USER command is sent, all other commands will receive the response:

-XXXX (104) Not logged in<NULL>

where "XXXX" is the command sent by the client.

Note: in this version of the protocol, the <user-name> is not validated but is required. Client applications should submit any name they wish conforming to the above specification. Failure to include the user-name will cause the following error to be generated:

```
-USER (201) User name is required<NULL>
```

“PSWD” Command

The PSWD command is used to establish a level of security on the connection. Different connections may desire different services. The WNDPSVR program may require the application to provide a User name and a password before certain services are allowed.

The format of the command is:

```
PSWD <password><NULL>
```

The <password> is assumed to be all characters following the single space character (which must follow the string “PSWD”) up to, but not including, the <NULL> character. The <password> may not include a <NULL> character. The passed password is the MD5 encoded bin64 version of the password (a description of the exact procedure will be included in a future version of the protocol specification).

If WNDPSVR accepts the password, it will respond:

```
+PSWD<NULL>
```

If WNDPSVR rejects the password, it will respond with:

```
-PSWD (300) Invalid password<NULL>
```

WNDPSVR requires that a client be successfully logged in before processing any other commands. The TCP connection will not be terminated, but unless a valid PSWD command is sent, all other commands will receive the response:

```
-XXXX (102) Not logged in<NULL>
```

where “XXXX” is the command sent by the client.

Note: this command is not currently implemented and is reserved for future use.

All PSWD commands will receive the following response:

```
-PSWD (103) Command not yet implemented<NULL>
```

"CLAS" Command

The CLAS command is used to instruct the WNDPSVR program to send messages based on the type (classification) of data contained in each message. The CLAS command appears as follows:

```
CLAS <classification-list><NULL>
```

<classification-list> is a space delimited list of one or more data classes. Example data classes are XMLNews-Story, HTML, ANPA and MarketData. (The CLAS command is not case-sensitive with respect to the elements of <classification-list>.) To provide upward compatibility, class names may not begin with a digit.

When a connection to WNDPSVR is established, the CLAS state is to send only XMLNews-Story messages. It is as though the client application had sent the command:

```
CLAS XMLNEWS-STORY<NULL>
```

If WNDPSVR accepts a CLAS command, it will respond with:

```
+CLAS<NULL>
```

When WNDPSVR accepts a CLAS command, the classes specified in that command **replace** the current setting (if a previous setting has occurred) or the session default setting. Optionally, specify the ALL class to disable suppression of data based on its class. (i.e. All data will be sent regardless of class type.)

If WNDPSVR rejects a CLAS command, it will respond with one of the following:

```
-CLAS (901) Missing class specification<NULL>
```

```
-CLAS (902) Invalid class specification<NULL>
```

When WNDPSVR rejects a CLAS command, the session will be maintained with the current setting (if a previous setting has occurred) or the session default setting.

Note: The class of each message will be indicated in the response to each RQST command.

"FLTR" Command

The FLTR command is used to instruct the WNDPSVR program to keep or discard messages based on their provider or their provider/service. The FLTR command appears as follows:

```
FLTR <filter-action> <provider> [<service>]<NULL>
```

<filter-action> is one of the keywords INCLUDE or EXCLUDE, <provider> is a provider string (as it appears in a story identifier), and <service> is a service provider string (as it appears in a story identifier). The <service> specification is optional. When it is omitted, the appropriate action is taken for all services of the specified provider. The special keyword ALL can be used in place of <provider>. In that case <service> is ignored.

When a connection to WNDPSVR is established, the FLTR state is to send all messages. It is as though the client application had sent the command:

```
FLTR INCLUDE ALL<NULL>
```

Providers and services can be excluded from the message stream by sending the required FLTR EXCLUDE <provider> [<service>] commands. (If a FILTER INCLUDE <provider> [<service>] command is sent after the corresponding FILTER EXCLUDE command, the appropriate messages will again be received by the client application.)

Similarly, the command:

```
FLTR EXCLUDE ALL<NULL>
```

can be used to specify that no messages are desired. Then the desired providers and provider/services can be added one-at-a-time using the appropriate FLTR INCLUDE commands.

If WNDPSVR understands the filter restriction and can accept it, it will respond with:

```
+FLTR<NULL>
```

If WNDPSVR cannot understand or comply with the filter request, then it will respond with one of the following:

```
-FLTR (700) Invalid filter method 'XXXX'<NULL>
```

```
-FLTR (701) Missing provider/service specification<NULL>
```

When WNDPSVR rejects a filter command, the session will be maintained with the current setting (if a previous setting has occurred) or the session default setting.

"FROM" Command

The FROM command is used to instruct the WNDPSVR program to begin sending data from the requested point in time. This can be used for client recovery after being stopped and restarted or if all client-side data needs to be recovered from the server.

The format of the command is:

```
FROM yyyymmdd hhmmss<NULL>
```

where:

yyyy – four-digit year

mm – two-digit month (01-12)

dd – two-digit day (01-31)

hh – two-digit hour (00-23)

mm – two-digit minute (00-59)

ss – two-digit second (00-59)

The WNDPSVR program responds with:

```
+FROM yyyymmdd hhmmss<NULL>
```

There are three successful variations to the response based on the available data. When the specified date/time parameters correspond to an available time frame, the WNDPSVR response will exactly match that of the request. When the specified date/time parameters correspond to a time frame that is earlier than the oldest available data, the response will contain the oldest available yyyymmdd with time hhmmss. When the specified date/time parameters correspond to the latest available day, but the time is in the future, the response will contain the latest available date and time.

It is an error to request a date that is greater than the most current day. The response will be:

```
-FROM (503) Requested data not yet available<NULL>
```

NOTE: If the server is restarting after longer periods of downtime, it is possible that the latest available news date may not be equal to the current day. Client applications may then wish to issue the RQST command without issuing a FROM command so that they will receive the most recently available data.

Design Note

The historical depth of retrievable information will be configured based on server hardware constraints. This depth can range from minutes to several months of data.

"RQST" Command

The RQST command is used by the client to obtain the name(s) of the next available message(s) along with their corresponding file sizes (in bytes). The file size is separated from the file name using slash as the delimiter. Using a RQST command without first submitting a FROM command will position the client application to the end of all currently available data. The client will therefore receive only new data that becomes available.

The RQST command is sent by the client and looks like:

```
RQST<NULL>
```

If the client application is not requesting messages quickly enough, an error may be returned when trying to read the corresponding data file. The time frame in which data must be received by the client from the server may range from minutes to days. This interval is dependent on the server configuration. Client application programmers will need to contact their system administrator in order to obtain the appropriate information.

Error message 400 signals the client application that it is not taking messages fast enough and that messages are being lost. The client application program should log this and notify the appropriate development personnel that there is a problem.

The WNDPSVR program may respond in one of two ways. If there are no messages currently available and none are received within 8 seconds, the WNDPSVR No content program will return the response:

```
-RQST (600) <NULL>
```

This occurs when there are no messages available within the timeout period. The client machine should reissue the RQST.

When no data is available, WNDPSVR program will respond with:

+RQST yyyymmdd hhmmss

NOTE: The *timestamp* (yyyymmdd hhmmss) is only returned when the client requests version 4 or greater. For more information please reference the "VRSN" command below.

When available files(s) exist, the WNDPSVR program will respond with:

+RQST yyyymmdd hhmmss class data-file/file-size [<associated file(s)>]<NULL>

where:

yyyy – four-digit year

mm – two-digit month (01-12)

dd – two-digit day (01-31)

hh – two-digit hour (00-23)

mm – two-digit minute (00-59)

ss – two-digit second (00-59)

class – data classification of the files referenced

It is the responsibility of the client to store the file name(s) and individually request each file they wish to receive using the FILE command.

Design Note

The reason for the 8 second timeout on the RQST message (i.e., when WNDPSVR returns the "-RQST" response) is to ensure that there is always traffic going between the client and the server. This validates the integrity of the hardware connection between client and server. This insures that the client can detect the loss of the TCP connection and log the appropriate warning.

Note that the requester should wait for 12 or more seconds to timeout even though the WNDPSVR program is to respond in 8 seconds. This is to allow for transit time between the WNDPSVR program and the client. Since the client and the WNDPSVR program are on the same box or same LAN segment, this timing should be adequate.

“FILE” Command

The FILE command is used by the client to request the contents of a single data file. The data file names are received via the RQST command. It is up to the client application to parse the returned file names and then issue an appropriate FILE command for each file they wish to receive. If the client is not interested in any of the files returned from the RQST command, they would simple issue another RQST to advance to the next set of files.

FILE data-file<NULL>

In the normal sequence for the RQST/FILE commands, the basic flow of messages for a single file transfer is:

Client -> WNDPSVR	WNDPSVR -> Client
RQST	
	+RQST yyyymmdd hhmmss class data-file/size
FILE data-file	
	+BLK0:bbbb{bbbb binary characters}BLKm
	+BLK1:bbbb{bbbb binary characters}BLKm
	+BLK2:bbbb{bbbb binary characters}BLKe

The description of the above is as follows. Initially, the client sends out a RQST command. This is a request for the name(s) of the next file(s) to be provided to the client.

The WNDPSVR program responds with:

+RQST yyyymmdd hhmmss class data-file/file-size [assoc. file(s)]

The string “+RQST” is the response indicating that the WNDPSVR program has one or more files to send.

The client program responds:

FILE data-file<NULL>

indicating to send the data file.

The WNDPSVR program will then begin sending the blocks of data. A block looks like:

+BLK***nn***:bbbb{bbbb binary characters}BLK***t***

This block consists of the characters "+BLK" followed by an incrementing ascii numeric value ("***nn***" above) followed by a colon (the first block will be numbered block "0", the next block is block "1", and so on). The characters after a colon until the ending "BLK" delimiter are all binary. The four bytes immediately following the colon are the four-byte length of the characters that are being sent. These four bytes are in network order (i.e., high-order byte first, low-order bytes after). These four bytes specify the number of bytes that will follow (the length bytes bbbb do not include the BLK delimiter at the end). Once the specified number of binary bytes has been sent, there is a delimiter sequence "BLK". Finally, there is block type character ("t") which indicates whether this is the last block ("e" = end) or not ("m" = more). Note that the curly brackets shown above are not sent (i.e., the data begins immediately after the four bytes of length are sent).

Design Note

After some discussion, it was decided that it would be best to limit the FILE command to request only one file at a time. There are error conditions that would create an ambiguous condition if a client were to submit multiple filenames some of which were valid and others of which were not.

"DSTR" Command

This command is only available in version 5 or later.

The DSTR command is used by the client to obtain the name(s) of the next available message(s) along with their corresponding data stream.

Using a DSTR command without first submitting a FROM command will position the client application to the end of all currently available data. The client will therefore receive only new data that becomes available.

The DSTR command is sent by the client and looks like:

```
DSTR<NULL>
```

Error message 400 signals the client application that it is not taking messages fast enough and that messages are being lost. The client application program should log this and notify the appropriate development personnel that there is a problem.

The WNDPSVR program may respond in one of two ways. If there are no messages currently available and none are received within 8 seconds, the WNDPSVR program will return the response:

+DSTR yyyymmdd hhmmss bbbb

When bbbb is 0000, which means there are no messages available within the timeout period. The client machine should reissue the DSTR.

When available files(s) exist, the WNDPSVR program will respond with:

+DSTR yyyymmdd hhmmss bbbb class file-name [<associated file(s)>file-size{binary bytes}file-size{binary bytes}…]

where:

yyyy – four-digit year

mm – two-digit month (01-12)

dd – two-digit day (01-31)

hh – two-digit hour (00-23)

mm – two-digit minute (00-59)

ss – two-digit second (00-59)

bbbb – 4-byte network byte-order unsigned integer presenting the following data length for class and file names.

class – data classification of the files referenced

file-name – the file names

file-size – 4-byte network-byte unsigned integer presenting the following file data length.

It is the responsibility of the client to store the file name(s) and the file content.

The sequence of file data is in the same order of file list.

Design Note

The reason for the 8 second timeout on the DSTR message (i.e., when WNDPSVR returns the “+DSTR yyyymmdd hhmmss bbbb” response) is to ensure that there is always traffic going between the client and the server. This validates the integrity of the hardware connection between client and server. This ensures that the client can detect the loss of the TCP connection and log the appropriate warning.

Note that the requester should wait for 12 or more seconds to timeout even though the WNDPSVR program is to respond in 8 seconds. This is to allow for transit time between the WNDPSVR program and the client. Since the client and the WNDPSVR program are on the same box or same LAN segment, this timing should be adequate.

"VRSN" Command

The VRSN command is used to specify which version of the WNDP protocol the client application is going to use. This command needs to be given only if the version number sent by the server when the TCP connection is established does not match the version of the protocol that the client wishes to use. Note: VRSN should be the first command sent after the TCP connection is established.

```
VRSN version<NULL>
```

If WNDPSVR supports the requested version, it will respond:

```
+VRSN<NULL>
```

If WNDPSVR does not support the requested version, it will respond with:

```
-VRSN (800) Protocol version not supported<NULL>
```

NOTE for VRSN 4: version 4 protocol includes a timestamp with the RQST answer when no new data has arrived.

Client -> WNDPSVR	WNDPSVR -> Client
VRSN 4	
	+VRSN
RQST	
	+RQST yyyymmdd hhmmss

The timestamp returned in a RQST response should be saved by the client for use on startup. Please refer to the "FROM" command for more information.

“CNFG” Command

The CNFG command is used to set configuration parameters.

Note: this command is not currently implemented and is reserved for future use.

All CNFG commands will receive the following response:

-CNFG (103) Command not yet implemented

Example Conversation

The following is an example of the conversation between the WNDPSVR program and the client. It shows the user logging in, sending an XML file, sending an associated JPEG file, and then requesting the next available set of data.

Client -> WNDPSVR	WNDPSVR -> Client
	+WAVO WNDP v3.00.00<NULL>
USER username <NULL>	
	+USER<NULL>
RQST<NULL>	
	+RQST yyyymmdd hhmmss class abc.xml/4038 xyz.jpg/15773<NULL>
FILE abc.xml<NULL>	
	+BLK0:bbbb{bbbb binary characters}BLKe
FILE xyz.jpg<NULL>	
	+BLK0:bbbb{bbbb binary characters}BLKm
	+BLK1:bbbb{bbbb binary characters}BLKm
	+BLK2:bbbb{bbbb binary characters}BLKe
RQST<NULL>	
	+RQST yyyymmdd hhmmss

Change History

5.00.00

- Document DSTR command.

4.00.00

- VRSN 4 protocol enhancement.

3.00.01

- Additional CLAS types (HTML, ANPA) added.
- CLAS command allows ALL option.

3.00.00

- Document CLAS command.
- Document change in RQST output due to data classes.
- Corrected text of error 700 (FLTR command).
- Add error 800 (VRSN command) to error number table.

2.00.04

- Document VRSN command.
- Removed date/time specification from FILE command.

2.00.03

- Add file size information to response to RQST command.

2.00.02

- Document FLTR command.
- Fix minor errors in example conversation.

2.00.01

- Document format change in the form of a tutorial followed by the technical specification (Appendix A).

- Version number response from WNDP server now returns the protocol version number only.
- When requesting a date using the FROM command that is earlier than the oldest available data, the response will contain the first available date (yyyymmdd) and corresponding time (hhmmss). In version 2.00.00, the time was 000000.

2.00.00

- Incoming data FILE(s) from server no longer require ACK from client.
- Renamed to Web Network Distribution Protocol (WNDP).
- Upon successful connection, the response no longer includes the version number of the server.
- Removed all references and implementation regarding the connection ID [nn:].
- HRTB and DONE commands no longer supported.
- New commands FROM and FILE.
- Revised transaction protocol for file request and transfer (RQST/FILE).
- Blocksize increased from 2 to 4 bytes (BLK command).
- All error messages have been revised.

1.00.01

- Changed “XML” to “BLK”.
- Changed timeout to 8 seconds.
- Attempted to make error responses more consistent.

1.00.00

- Initial release.

Error Numbers

Number	Context	Explanation
100	Initialization	Service unavailable
101	-UNKN	Bad request 'xxxx'
102	-XXXX	Server error
103	-PSWD -CNFG	Command not yet implemented
104	-XXXX	Not logged in
200	-USER	Invalid user name 'xxxx'
201	-USER	User name is required
300	-PSWD	Invalid password
400	-FILE -RQST -DSTR	Data file ‘xxxx’ does not exist
401	-FILE	Cannot open data file 'xxxx'
402	-FILE	Requested file has not been updated
500	-FROM	Invalid date 'xxxx'
501	-FROM	Invalid time 'xxxx'
502	-FROM -FILE	Missing date/time specification
503	-FROM	Requested data not yet available
600	-RQST	No content
700	-FLTR	Invalid filter method ‘xxxx’

701	-FLTR	Missing provider/service specification
800	-VRSN	Protocol version not supported
901	-CLAS	Missing class specification
902	-CLAS	Invalid class specification

NOTE: All above references to “xxxx” will be filled in dynamically based on the context in which the error condition occurs.