Avid Technology, Inc.

Avid iNEWS® NRCS

iNEWS FTP/FTPS Server Protocol Specification

Version 2017.1

25 September 2017

1 Introduction

The iNEWS FTP Server enables any network client to acquire stories and lists of stories from iNEWS via FTP (File Transfer Protocol) interaction. The iNEWS FTP Server supports a subset of the FTP commands described in the FTP specification and implements some extensions. This document describes the supported commands, the iNEWS specific extensions, and the format of some of the data returned by some commands.

Standard FTP services are available at TCP port 21 by default. The iNEWS FTP service may instead be available on a different port, depending on the site configuration. The FTP client must be prepared to initiate a connection to any valid TCP port.

The 1.5.1 iNEWS FTP Server conforms to Version 2.4 of this specification.
The 2.1.5 iNEWS FTP Server conforms to Version 2.6 of this specification.
The 2.5.2 iNEWS FTP Server conforms to Version 2.8 of this specification.

The 3.0.0 iNEWS FTP Server conforms to Version 3.0 of this specification.
The 3.3.0 iNEWS FTP Server conforms to Version 3.1 of this specification.
The 3.5.0 iNEWS FTP Server conforms to Version 3.5 of this specification.

The 2017.1 iNEWS FTP Server conforms to Version 2017.1 of this specification.

1.1 Changes from 2.6 to 2.8

The primary change is to use Unicode as the native iNEWS Server character set. The SITE CHARSET command was introduced to allow for compatibility with legacy iNEWS systems.

The ***FEAT*** and ***OPTS*** commands were added.

The ***SITE FAST*** value can no longer be ***0***. This value will disable the use of the ***FAST*** protocol.

The ***Story Exchange Protocol (SEP)*** format is no longer supported.

The ***SITE LISTSZ*** default value is now 500.

The ***SITE SENDFORM*** command was introduced to allow the LOOK section of stories to be sent and received.

1.2 Changes from 2.8 to 3.0

Added support for NSML 3. This format is identified by "3nsml."

1.3 Changes from 3.0 to 3.1

Added support for NSML 3.1. This format is identified by "3.1nsml."

1.4 Changes from 3.1 to 3.5

The Block Transfer Mode is now supported. See the ***MODE*** command for details.

1.5 Changes from 3.5 to 2017.1

FTPS is now supported. See the ***FTPS Command Set***, ***FTPS Create Certificates***, ***FTPS RXnet/TXnet***, and ***FTPS Replies*** for more details.

1.6 Referral Documents

Story documents retrieved from iNEWS are in News Story Markup Language (NSML) format. NSML is an SGML-based markup language described in "NSML – A Markup Language for News Stories," by iNEWS.

This document does not describe the FTP protocol. The FTP protocol is described in "Network Working Group Request for Comments: 959," by J. Postel and J. Reynolds (http://www.ietf.org/rfc/rfc959.txt). The iNEWS FTP server conforms to this specification unless otherwise stated in this document.

2 FTP Command Set

The following standard set of FTP commands will be recognized by the iNEWS FTP server.

2.1 CWD [<pathname>]

Use CWD to go to the indicated queue or directory name. The pathname is in standard iNEWS format: each directory level name is separated by a dot. Names are case insensitive. To indicate a queue or directory relative to the current directory, place a dot before the name. For example:

CWD show.8PM.rundown — go to ***show.8pm.rundown***
CWD .8pm.rundown — if the current directory is ***show*** go to ***show.8pm.rundown***

If no ***pathname*** is supplied, the user's default destination is used.

The parent directory of the current directory is denoted with a ***pathname*** of "**..**"

Note: Some FTP clients use the forward slash character, '/', when constructing ***pathnames***. To facilitate the use of such FTP clients, when the iNEWS FTP Server encounters a ***pathname*** that starts with a forward slash, the iNEWS FTP Server will remove forward slashes from the ***pathname***. Specifically, a leading forward slash, any forward slash that precedes a period, '.', and a trailing forward slash will be removed from the ***pathname***. The following commands are equivalent to the ones shown above and will produce identical results:

CWD /show/.8PM/.rundown/
CWD /.8pm/.rundown/

2.2 DELE <name>

Use DELE to remove the indicated story from the current queue provided the queue has the UPDATE queue trait. The parameter to the DELE command is a **<name>** from a story list entry returned by the LIST or NLST commands. Only the **<story identifier>** part is used. Everything past the first colon in **<name>** is discarded.

For example, the following DELE commands are all valid and remove the same story:

DELE 067801F2

DELE 067801F2:0852002F:12642A3B

DELE 067801F2:0852002F:12642A3B Story Name

If the current queue does not have the UPDATE queue trait, the FTP Server will ignore the request and respond with:

```
200 DELE command successful.
```

2.3 FEAT

This command reports the extended features the server supports. Currently only one extended feature is supported.

The server will respond with:

```
211-Extensions supported:
 UTF8
211 End
```

2.4 HELP

The server will respond with:

```
214 I cannot help you.
```

2.5 LIST

The LIST command returns a list of the directories or queues in the current directory, or a list of the stories in a queue. Each list entry contains attributes of the directory, queue, or story This command deviates from the standard FTP command in that it will ignore pathname parameter if one is supplied.

Each entry in the listing will be a single line containing the following information:

<flags> <group id> <owner> <length> <modification date> <name>

Each item is separated by one or more spaces. The ***modification date*** consists of three "words", the month, the day, and the time or year. The ***modification date*** will have the time if the data is within the last six months, otherwise it will have the year. There will always be 7 "words" preceding the ***name***.

2.5.1 Directory Listing

Listing a directory will produce an entry for each directory or queue contained in it. The first character of the **<flags>** will be "d," and the **<name>** will be the directory or queue name. All the other fields in the entry have no real meaning and are included only for compatibility with standard Unix style directory listings. The **<owner>** will be a "d" or a "q" to indicate that the entry is a directory or a queue. The **<length>** will be the number of sub-directories contained in the directory if the entry is a directory. If the entry is a queue, **<length>** will be 0. All names are preceded by a period to indicate that the name is relative to the current directory.

For example:

```
d-------- 1  d   5 Feb 20 14:05  .A_Directory
d-------- 1  q   0 Feb 20 14:05  .A_Queue
```

In this case:

```
<flags>                 = d--------
<group id>              = 1
<owner>                 = d for the directory, q for the queue
<length>                = 5 subdirectories in directory, 0 for a queue
<modification date>     = Feb 20 14:05
<name>                  = A_Directory / .A_Queue
```

2.5.2 Queue Listing

Listing a queue will produce entries for stories contained in it. The first character of the **<flags>** will be a dash, and the **<name>** will be a unique story identifier composed as follows:

<story identifier>:<story locator> <story name>

The **<story identifier>** is unique within a queue listing. It is constant across story edit changes, so it can be

used to decide if a change is the result of an addition to the queue or an edit of a previously existing story. It is an eight-character string.

The **<story locator>** is a unique identifier used to locate a story in the iNEWS database. It consists of two eight-character strings separated by a colon. Unlike the story identifier the story locator will change each time a story is edited on iNEWS.

The **<story name>** is separated from the **<story locator>** by a space. It is an arbitrary string of characters that may include spaces.

The second character of the **<flags>** field will be a lower case "f" if the story is "floating" in the rundown. A story is marked as floating when it has not yet been committed to the rundown.

The **<length>** is the duration of the story expressed in seconds.

The **<modification date>** is the date and time of last modification. If the last modification was executed more than six months ago, the time of day will be replaced with the year, for example "`Jun 20 1996.`"

The **<group id>** and **<owner>** have no real meaning and are included only for compatibility with standard Unix style directory listings.

For example:

```
-f-------- 1 1 38 Feb 20 14:05 067801F2:0852002F:12642A3B Story Name
```

In this case:

```
<flags>                = -f--------
<group id>             = 1
<owner>                = 1
<length>               = 38
<modification date>    = Feb 20 14:05
<name>                 = 067801F2:0852002F:12642A3B Story Name
```

By comparing the differences between two consecutive LIST commands, five types of changes may be revealed: story deleted, story added, story modified, story float status changed, and list order changed. When a story is modified, both the modification date and the **<story locator>** will change. It is recommended that the **<story locator>** be used to determine if a modification has occurred. The **<story locator>** is a much more reliable and accurate indicator for story modification than the **<modification date>**. Any number of changes may occur between two LIST commands. A limit is placed on the number of stories that will be listed by the LIST command to prevent absurdly long listings. The default limit is 500 stories. This limit can be set by the FTP client via the SITE LISTSZ command (see 2.17.6).

2.5.3 User Account Listing

A variant of the LIST command can be used to get a list of user account information for one or more users. To request user account information use:

LIST .users.[<pattern>]

When a parameter is included with the LIST command and the parameter begins with the 7 character string **".users."**, the FTP server will attempt to provide a list of user account information for all users that match the <**pattern**>. The **<pattern>** is optional and if missing implies all users. The **<pattern>** can include a trailing asterisk as a wildcard. So a **<pattern>** of a single asterisk is equivalent to a missing **<pattern>**. To get user account information for all users starting with 'a' the **<pattern>** would be "a*".

The user account information listed is:

<user> su=<yes | no> b=<yes | no> vb= <yes | no> realname=<user's real name>

<user> is the user's login name; **"su"** represents the **superuser** user attribute; **"b"** represents the **blacklist** user attribute; **"vb"** represents the **video browse** user attribute.

2.6 MKD <type>.<name>

Use this command to make (create) a directory or queue within the current directory. The **<type>** parameter is either'd' (or 'D') or 'q' (or 'Q') to indicate that a directory or a queue is being created respectively. A period, '.', immediately follows the **<type>** letter, and the **<name>** immediately follows the period. No spaces are allowed between the **<type>,** the period, and the **<name>**.

<name> can be at most 20 characters and cannot include a period.

This command will also fail if:

- The "current directory" is a queue.
- The user does not possess the "enter & remove" attribute.
- The user does not have write group access to the "current directory". The server will report a successful operation with:

  `200 MKD <type>.<name> command successful.`

2.7 MODE <type>

This command sets the transfer mode for data transfers. Recognized types are ***Stream*** and ***Block***.

All other types will result in a "501 … invalid parameter" response.

Block mode is incompatible with the non-standard ***FAST protocol.***

When using Stream mode, each **LIST, NLST, RETR,** and **STOR** command results in a new TCP connection being used for each data transfer. The closing of the data connection indicates the end of file for the data transfer.When using Block mode, a single TCP connection is made and all data transfers are made through that connection. The data stream contains block information that denotes the end of file. This allows the FTP server to reuse the single TCP data connection.

2.8 NLST

The NLST command returns a name list of the directories or queues in the current directory, or a list of the stories in a queue. This command deviates from the standard FTP command in that it does not take a pathname parameter.

2.8.1 Directory Listing

Listing a directory will produce an entry for each directory or queue contained in it. The entry will consist only of the name preceded by a period.

For example:

.DEAD
.FUTURES
.PEOPLE
.SHOW
.SYSTEM
.TEST
.WIRES

2.8.2 Queue Listing

Listing a queue will produce entries for stories contained in it. Each entry will consist only of the first 8 hexadecimal digits of the story id which is suitable for use in the **RETR** command.

For example:

11A1313F
12A1313F
13A1313F

1BA1361E
1CA1361E
1DA1361E
1EA1361E
1FA1361E
00A1361F
01A1361F
04A1361F
05A1361F
06A1361F
08A1361F
0AA1361F
0BA1361F

A limit is placed on the number of stories that will be listed by the NLST command to prevent absurdly long listings. The default limit is 500 stories. This limit can be set by the FTP client via the SITE LISTSZ command (see 2.17.6).

2.8.3 User Name Listing

The "**NLST .users."** command is similar to the **"LIST .users."** command described above. Each line of the listing produced only includes the user name. No other account information is included.

NOTE: An earlier version of this specification had the NLST command being an alias for the LIST command. Some FTP clients depend on that behavior. It is possible to have the iNEWS FTP Server treat the NLST command as an alias for the LIST command. This is accomplished by setting "RXOLDNLST=1" in the iNEWS FTP Server program environment.

2.9 NOOP

The server will respond with:

```
200 NOOP command successful.
```

2.10 OPTS <command> [<command-options>]

This command allows the client to select the behavior extended features. Currently only one extended feature is supported:

UTF8 [on]

When the server receives a "**OPTS UTF8 on**" command it will respond with:

```
200 OPTS UTF8 command successful.
```

If an unrecognized command is received the server will respond with:

```
501 OPTS <command> - invalid parameter.
```

2.11 PASS <password>

This command supplies a password for the user named in the USER command. It must immediately follow the USER command.

The server will report a successful login with:

```
230 User <user name> logged in.
```

If there isn't an account for the user or the supplied password is not correct, the server will respond with:

```
530 Login failed for <user name>.
```

2.12 PASV

This command requests the server to "listen" on a data port (which is not its default data port) and to wait for a connection rather than initiate one upon receipt of a transfer command. The response to this command includes the host and port address this server is listening on in the format defined in the PORT command, for example:

```
227 Entering Passive Mode. (h1,h2,h3,h4,p1,p2)
```

2.13 PORT h1,h2,h3,h4,p1,p2

This command can be used to specify the data port to be used when transferring data with commands LIST, RETR, and STOR. The 32-bit internet host address is specified by 4 8-bit fields, ***h1-4*** and the 16-bit TCP port address by 2 8-bit fields, ***p1-2***. All fields are decimal numbers (in character string representation. The fields are separated by commas. Fields are ordered high to low.

The specified port will be used until changed by another PORT command.

2.14 PWD

This command will report the current working directory name.

2.15 QUIT

The server will close the connection after responding with:

```
221 Goodbye.
```

2.16 RETR <name>

Use RETR to retrieve the indicated story from the current queue. The parameter to the RETR command is a **<name>** from a story list entry returned by the LIST or NLST commands. If the story indicated by **<name>** is not in the current queue, the FTP error code 550 is returned. Only the **<story identifier>** part is used. Everything past the first colon in **<name>** is discarded.

For example, to retrieve the above story, any of the following RETR could be issued:

RETR 067801F2

RETR 067801F2:0852002F:12642A3B

RETR 067801F2:0852002F:12642A3B Story Name

Stories are formatted in either the NSML or 2NSML format. You can use the SITE FORMAT command to determine the format.

Warning: if the current format is something other than NSML or 2NSML, NSML will be used.

Warning: The look section of the story is NOT included.

NOTE: Some FTP clients include the current working directory as a prefix to **<name>**. To facilitate the use of such FTP clients, the iNEWS FTP Server will attempt to strip off this prefix. The rules used to determine if the **<name>** parameter includes the current working directory as a prefix and to strip off this are:

- If the first character is a forward slash, discard the forward slash and consider the next character to be the first character.
- If the first character is a period, discard the period.
- Match the **<name>** parameter against the current working directory. Consider a forward slash and a forward slash followed by a period to be a match with the iNEWS standard separator, namely, the period. If the entire current working directory matches the **<name>** parameter, discard all matched characters of **<name>**.
- Strip off a trailing forward slash.
- Strip off a trailing period.

- The next 8 characters are used as the **<story identifier>**.

As an example, the following commands would be equivalent to the above commands when the current working directory is **show.8pm.rundown**:

RETR /show/.8pm/.rundown/067801F2

RETR /show/8pm/rundown/.067801F2:0852002F:12642A3B

NOTE: FTP clients typically use a line of the output of the NLST command as the **<name>** parameter. Since this is simply the **<story identifier>**, no problems should be encountered. However, when the iNEWS FTP Server has been configured to use the "OLD NLST" behavior (See the note at the end of Section 2.8), the iNEWS FTP Server will attempt to match and discard text that matches the initial portion of a LIST output line from the **<name>** parameter. This is intended to preserve functionality for FTP clients following an earlier version of this specification.

2.17 SITE

The client may send this command to view or change parameters specific to the iNEWS implementation of FTP. Only a single parameter should be used within a SITE command. When a parameter does not include the optional setting, the iNEWS FTP server will report the parameter's current setting.

If multiple parameters are supplied and any of the parameters not recognized or formatted correctly, the iNEWS FTP Server will reject the entire command and return FTP error code 501.

2.17.1 SITE CHARSET [= <charset name>]

This is used to specify the character set used by the FTP client. All characters received from the client will be converted from this character set into Unicode and all characters sent to the client will be converted from Unicode into this character set.

The list of valid character set names can be produced by using the ***iconv –list*** Linux console command.

Conversions are performed on all characters sent and received on the FTP command connection as well as characters sent and received on the FTP data connection.

When a character received from the FTP client cannot be converted to Unicode, a hexadecimal representation of the character preceded with a backslash (**"\hh"**) is used as a replacement of the character.

When a Unicode character to be sent to the FTP client cannot be converted from Unicode, the sequence **U+hhhh** will be used as a replacement for the Unicode character.

The iNEWS FTP Server can be configured to use a specific character set by setting "RXSITECHARSET=<new charset>" in the iNEWS FTP Server program environment. This is useful when the FTP client cannot submit a SITE CHARSET command.

2.17.2 SITE FAST [= <number>]

NOTE: Use of the ***FAST protocol*** is deprecated. Block transfer mode should be used instead. There can be network equipment in the path between the FTP server and client that prohibits data transfers over the FTP command connection. See the MODE command (2.8) for more information.

FAST instructs the server to NOT create a new TCP data stream for each LIST, NLST, RETR, and STOR as required by the FTP standard. Instead data will be transferred over the same connection used by the FTP commands. The data streams will be terminated with an end-of-file mark. This command specifies a decimal integer between 1 and 255 to use as the end-of-file mark.

Setting FAST to zero causes the standard FTP behavior, that is, a new data stream connection is made for each data transfer.

Use of the non-standard FAST protocol can be disabled by settting "RXNOFAST=1" in the iNEWS FTP

Server program environment. This is useful for iNEWS to iNEWS FTP connections that must go through firewalls when using the standard FTP command port (port 21). Many firewalls will prevent the sending of non-FTP command data through that connection.

2.17.3 SITE FORMAT [= <format name>]

FORMAT is used to specify the format to be used when sending stories to the client. This parameter can be set to ***nsml, 2nsml, 3nsml, or 3.1nsml.*** The default is NSML.

The **2nsml** format is a superset of the **nsml** format. It includes story attachments.

The **3nsml** format is well formed XML and includes many new tags as well as removals and renaming of existing tags.
The **3.1nsml** format is a superset of the **3nsml** format. It includes the ***projects*** story section.
The iNEWS FTP Server can be configured to use a different format by setting "RXSITEFORMAT=<new format>" in the iNEWS FTP Server program environment. This is useful when the FTP client cannot submit a SITE FORMAT command.

2.17.4 SITE HELP

This command will report the SITE commands supported.

2.17.5 SITE IDLE [= <seconds>]

IDLE is used to specify the maximum number of seconds the connection is allowed to be idle before closing the connection. If set to 0, no idle timeouts will occur. This parameter is set to the iNEWS system ***localtimeout*** parameter. Note, that this is a decimal number of seconds, so "905" represents 15 minutes and 5 seconds.

The iNEWS FTP Server can be configured to use a different value by setting "RXSITEIDLE=<new timeout>" in the iNEWS FTP Server program environment. This is useful when the FTP client cannot submit a SITE IDLE command.

2.17.6 SITE LISTSZ [= <number>]

LISTSZ is used to change the story limit used by the LIST command. Valid values are greater than or equal to 1. The default is 500.

The iNEWS FTP Server can be configured to use a different value by setting "RXSITELISTSZ=<new list size>" in the iNEWS FTP Server program environment. This is useful when the FTP client cannot submit a SITE LISTSZ command.

2.17.7 SITE SENDFORM [= < 0 | 1 >]

This command determines if the LOOK section of a story is included in stories RETRieved by the FTP client. The LOOK story section is excluded when SENDFORM=0 and is included when SENDFORM=1. The default setting is SENDFORM=0.

The iNEWS FTP Server can be configured to send the LOOK section by setting "RXSITESENDFORM=1" in the iNEWS FTP Server program environment. This is useful when the FTP client cannot submit a SITE SENDFORM command.

2.18 STAT

This command will report the iNEWS FTP Server status. This includes the program version, the name of the logged in user, and the values of the site parameters. Here is a sample response:

```
211-WINEWS-A NEWS Rxnet (ftp server) status:
    Version 2.5.2.41 LINUX.
    Logged in as inews.
    FORMAT=nsml
    LISTSZ=500
    IDLE=0
    CHARSET=UTF-8
211 End of status
```

2.19 STOR [order]

Use this command to send stories or a queue order list to the server. If the optional parameter "order" is specified, the server will expect a list of story identifiers in the order that the stories in the current directory are to be ordered. Otherwise, the server will expect one or more stories formatted according to the currently selected format (SITE FORMAT).

A sample order list in NSML (or 2NSML) format is:

```
QSTAMP=3848463961
QSTAMP=4184008281
QSTAMP=4519552601
QSTAMP=4855096921
```

Please note that the server will only reorder UPDATE queues. The server will always respond to the STOR order command with:

```
226 Closing data connection; STOR successful.
```

When STORing stories into an UPDATE queue, the server will use the qstamp included within the story to identify which story is to be updated. In NSML and 2NSML format this is "<storyid>**16f04a3c**:0003ab69:3af71214"

Stories STORed in non-UPDATE queue are given a new qstamp and simply added to the queue.

2.20 STRU <file structure>

This command is accepted, but largely ignored. The server will send successful response to "STRU F". All other types will result in a "501 … invalid parameter" response.

2.21 SYST

This command will report the operating system of the server. The response will be:

```
215 UNIX Type: L8 (iNEWS News System)
```

2.22 TYPE <type>

This command is accepted, but largely ignored. The server will send successful response to "TYPE I", "TYPE A" and "TYPE A N". All other types will result in a "501 … invalid parameter" response.

LIST data is sent as ASCII with a CR-LF sequence terminating each line. All story data (for RETR and STOR) is sent as "Image" data formatted according to the currently selected SITE FORMAT (See 2.17.3).

2.23 USER <user name>

The <user name> must be a valid user on the iNEWS system. Access to the iNEWS database is determined by the access granted to the named user. The PASS command must be the next command received by the server. The server responds with:

```
331 Password required for <user name>.
```

If this command is received after a successful login has been accomplished, the server will respond with:

```
503 Bad sequence of commands.
```

2.24 XCWD [<pathname>]

Same as **CWD.**

2.25 XMKD <type>.<name>

Same as **MKD.**

2.26 XPWD

Same as **PWD.**

3 FTP Replies

Each command is listed with its possible replies.

CWD	250, 530, 550
DELE	200, 451, 530, 550
FEAT	211
HELP	214
LIST	226, 425, 451, 530, 550
MKD	200, 451, 501, 550
MODE	200, 501
NLST	226, 425, 451, 530, 550
NOOP	200
OPTS	200;451,501
PASS	230, 451, 503, 530
PASV	227, 421
PORT	200
PWD	257, 530
QUIT	221
RETR	250, 425, 451, 501, 530, 550
SITE CHARSET	200, 501
SITE FAST	200, 501
SITE FORMAT	200, 501
SITE HELP	214, 501
SITE IDLE	200, 501
SITE LISTSZ	200, 501
SITE SENDFORM	200, 501
STAT	211

STOR	226, 425, 451, 530, 550
STRU	200, 501
SYST	215
TYPE	200, 501
USER	331, 503

4 FTPS Create Certificates

Create certificate and add it to trusted.

4.1 Generate root certificate

openssl req -newkey rsa:2048 -nodes -keyout cakey.pem -x509 -days 365 -out cacert.crt -outform PEM

4.2 Create certificate request

openssl req -newkey rsa:2048 -sha256 -nodes -keyout s2.pem -out s2.csr -outform PEM

4.3 Sign request and create certificate

openssl ca -config ca.cnf -policy signing_policy -extensions signing_req -out s2.crt -infiles s2.csr

Before "sign request and create certificate" - make sure to create index.txt file so signed key will be recorded into it:

```
$ touch index.txt
$ echo "01" > serial.txt
```

4.4 Add root certificate to trusted

cp cacert.crt /etc/pki/ca-trust/source/anchors

sudo update-ca-trust

NOTE: On RHEL 6, you have to activate the system with update-ca-trust enabled after installing the update.

Note: Add root certificate should be performed only on TXNET server. If you create and signed in certificate on TXNET server, then on RXNET server perform the following steps:

1. *Copy s2.crt to some folder on RXNET server, for example /site/ssl/cert*
2. *Copy s2.pem to some folder on RXNET server, for example /site/ss*
3. *cat s2.crt >> s2.pem*
4. *Add environment variable /site/env/rxnet*

   "RXCERTFILE=/site/ssl/cert/s2.pem"
5. *Verify that connection use trusted certificate*

   $ openssl s_client -tls1_2 -starttls ftp -connect 172.22.202.112:21

   -----END CERTIFICATE-----

   subject=/C=UA/ST=Kyiv/L=Brovary/O=OD/OU=QA/CN=SERV6

   issuer=/C=UA/ST=Kyiv/L=Brovary/O=OD/OU=QA/CN=SERV6/emailAddress=jane.doe@wavd.com

   ---

4.4.1 Config File for CA

Here is a sample of the ca.cnf file:

```
HOME       = .
RANDFILE   = $ENV::HOME/.rnd

####################################################################

[ ca ]
default_ca = CA_default      # The default ca section

[ CA_default ]

default_days      = 700      # how long to certify for
default_crl_days  = 30       # how long before next CRL
default_md        = sha256   # use public key default MD
preserve          = no       # keep passed DN ordering

x509_extensions = ca_extensions    # The extensions to add to the cert

email_in_dn     = no         # Don't concat the email in the DN
copy_extensions  = copy      # Required to copy SANs from CSR to cert
base_dir         = .
certificate      = $base_dir/cacert.crt  # The CA certifcate
private_key      = $base_dir/cakey.pem   # The CA private key
new_certs_dir    = $base_dir       # Location for new certs after signing
database         = $base_dir/index.txt   # Database index file
serial           = $base_dir/serial.txt  # The current serial number

unique_subject = no # Set to 'no' to allow creation of
                    # several certificates with same subject.

####################################################################
[ req ]
default_bits                = 4096
default_keyfile             = cakey.pem
x509_extensions             = ca_extensions
string_mask                 = utf8only

####################################################################

[ ca_extensions ]

subjectKeyIdentifier        = hash
authorityKeyIdentifier      = keyid:always, issuer
basicConstraints            = critical, CA:true
keyUsage                    = keyCertSign, cRLSign

####################################################################

[ signing_policy ]

countryName                 = optional
stateOrProvinceName         = optional
localityName                = optional
```

```
organizationName            = optional
organizationalUnitName      = optional
commonName                  = supplied
emailAddress                = optional

#################################################################

[ signing_req ]

subjectKeyIdentifier        = hash
authorityKeyIdentifier      = keyid,issuer
basicConstraints            = CA:FALSE
keyUsage                    = digitalSignature, keyEncipherment
```

5 FTPS Command Set

The following standard set of FTPS commands will be recognized by the iNEWS FTP server.

5.1 AUTH

This command instruct the server to switch to secured session. The only acceptable parameter for the AUTH command is TLS.

5.2 PBSZ

This command instruct the server to set protection buffer size. The argument is a decimal integer representing the maximum size, in bytes, of the encoded data blocks to be sent or received during file transfer. The response to this command includes actual buffer size, for example:

200 PBSZ=0 command successful.

5.3 PROT

This command indicates to the server what type of data channel protection the client and server will be using. The following codes are assigned:

- C - Clear
- S - Safe
- E - Confidential
- P - Private

The server accepts only Clear and Private types. For more information look into rfc2228.

5.4 CCC

This command instructs the server to switch back from secured session to plain. Not supported in current implementation. The server will respond with:

534 Request denied for policy reasons.

5.5 ADAT, MIC, CONF, ENC

The server will respond with:

503 Bad sequence of commands.

5.6 FEAT

The server will respond with:

211-Extensions supported:
AUTH TLS
PBSZ
PROT
UTF8
211 End

6 FTPS Replies

Each command is listed with its possible replies.

AUTH	234, 501, 503, 534
PBSZ	200, 501
PROT	200, 501, 536
CCC	503, 534
ADAT	503
MIC	503
CONF	503
ENC	503

7 FTPS – RXNET/TXNET Support

This section provides information on environment variables and default behavior related to RXnet and TXnet support of FTPS.

7.1 RXNET Environment Variables

RXREQUIRESSL=<0|1>

When RXREQUIRESSL=0 is set in /site/env/rxnet, then rxnet will allow old clients (that do not support FTPS) to connect to it. By default RXREQUIRESSL set to 1 and rxnet accepts only FTPS compatible clients.

RXCERTFILE=<certificate_filename>

The RXCERTFILE lets you set an custom certificate file for secured connections. By default RXCERTFILE set to /site/ssl/server.pem. Certificate pem file should contain certificate and private key without password.

7.1.1 Default Behavior

RXNet require secured connection. It means that client before login must must establish secure session (look into rfc4217 for more information). Otherwise rxnet will refuse all other commands. This behavior can be changed with RXREQUIRESSL environment variable.

Here is an acceptable sequence of commands to login to server:

```
                Client                          Server
Control                 Data            Data                    Control
=================================================================
                                                                socket()
                                                                bind()
socket()
connect()       ----------------------------------------------------------> accept()
                <---------------------------------------------------------- 220
AUTH TLS        ---------------------------------------------------------->
                <---------------------------------------------------------- 234
TLSneg()        <----------------------------------------------------------> TLSneg()
PBSZ 0          ---------------------------------------------------------->
                <---------------------------------------------------------- 200
PROT P          ---------------------------------------------------------->
                <---------------------------------------------------------- 200
USER fred       ---------------------------------------------------------->
                <---------------------------------------------------------- 331
PASS pass       ---------------------------------------------------------->
                <---------------------------------------------------------- 230
```

Rxnet FAST mode not supported in secured session. Instead of this client can use BLOCK mode (for addition information look into rfc959 chapter 3.4.2).

7.2 TXNET Environment Variables

TXVERIFYCERT=<0|1>

When TXVERIFYCERT=0 is set in /site/env/txnet, then txnet will not verify server's certificate. It means that txnet can connect to servers with certificate that is not trusted. By default TXVERIFYCERT set to 1 and txnet will fail to connect to servers with untrusted certificate.

7.2.1 Default Behavior

By default txnet will try to establish secure session with server (look into diagram in default behavior for rxnet). If it fails txnet will continue in insecure session, in this way txnet stays compatible with old rxnet/ftp servers.

From 2017.1 version of iNEWS txnet will use BLOCK mode instead of FAST and passive data connection. FAST mode stays acceptable for transferring stories to previous versions of iNEWS systems, but user need to set it in script manually.

7.3 Notes and Limitations

- rxnet not verify certificate for secured active data connections.
- txnet use /site/ssl/server.pem certificate for secured active data connections.
- rxnet/txnet supports only TLS authentication mechanism (AUTH TLS).