the high-performance real-time implementation
of TCP/IP standards

Post Office Protocol Version 3 (POP3)

User Guide

Express Logic, Inc.
858.613.6640
Toll Free 888.THREADX
FAX 858.521.4259
www.expresslogic.com

Part Number: 000-1054
Revision 5.0

Contents

Chapter 1

Introduction to NetX POP3

The Post Office Protocol Version 3 (POP3) is a protocol designed to provide a mail transport system for small workstations to access Client maildrops on POP3 Servers for retrieving Client mail. POP3 utilizes Transmission Control Protocol (TCP) services to perform mail transfer. Because of this, POP3 is a highly reliable content transfer protocol. However, POP3 is does not provide extensive operations on mail handling. Typically, mail is downloaded and then deleted. IMAP4 is a more advanced (and complex) protocol than POP and described in RFC 1730.

NetX POP3 Requirements

Client Requirements

The NetX POP3 Client API requires a creation of a NetX IP instance and NetX packet pool. Because the NetX POP3 Client utilizes NetX TCP services, TCP must be enabled with the *nx_tcp_enable* call prior to using the NetX POP3 API on that same IP instance. The POP3 Client uses a TCP socket to connect to a POP3 Server on the Server's POP3 port. This is typically set at the well-known port 110, though neither POP3 Client nor Server are required to use this port.

The size of the packet pool in terms of packet payload and number of packets available is user configurable, and depends on the anticipated volume of mail transmission and size of mail message content. The packet size associated with the POP3 Server or Client can be any size but it is recommended that packet size stay within the Ethernet device MTU (mean transfer unit) size minus the NX_PHYSICAL_HEADER (14 bytes).

The NetX POP3 Client must temporarily store downloaded mail before spooling it to a mail message file. The POP3 Client can be created with byte and block pool for this purpose or can use other storage e.g. stack memory for storing mail message data. The size of these memory resources as well depends on the anticipated mail traffic and available resources. See **NetX POP3 Memory Allocation** elsewhere in this document for more information about using Client memory.

To transfer mail messages from temporary memory to hard disk, the POP3 Client application must define a mail spooler callback. See **NetX POP3 Client Callbacks** elsewhere in this document for more details.

Server Requirements

The NetX POP3 Server API also requires a creation of a NetX IP instance and NetX packet pool. Like the NetX POP3 Client, the NetX POP3 Server utilizes NetX TCP services, and TCP must be enabled with the *nx_tcp_enable* call prior to using the NetX POP3 API on that same IP instance. The POP3 Server is defaulted to listen on the well known port 110 for POP3 Client connection requests, though neither POP3 Client nor Server are required to use this port.

The NetX POP3 Server API requires several callback services for creating, reading and deleting Client maildrops. See **NetX POP3 Server Callbacks** elsewhere in this document for more details.

The NetX POP3 Server requires creation of a byte pool for dynamic memory allocation for processing a Client maildrop, although this does not involve the actual bulk of the mail message data. The size of the byte pool as well as the packet pool depends on the anticipated mail traffic for the Server and available resources.

NetX POP3 Server and NetX POP3 Client applications can work with non NetX POP3 applications. A NetX POP3 Client and Server can run as concurrent tasks on the same system. This is a typical configuration of most commercial mail servers, where a incoming and ready-to-send mail queues are set up for coordinating receiving and transmitting mail simultaneously.

NetX POP3 Client and Server Constraints

While the NetX POP3 Client and Server implement the RFC 1939 protocol, there are some constraints:

1. The NetX POP3 Client and Server do not support the AUTH command (see RFC 1734) but do implement APOP authentication using DIGEST-MD5 for the Client Server authentication exchange.

2. The NetX POP3 Client and Server do not support IP6 addressing.

3. Domain Name Server (DNS) is supported by NetX but not directly by POP3 Client and Server API services.

4. NetX POP3 Client and Server API support the basic set of POP3 commands recommended by RFC 1939, but not the TOP or UIDL commands although the host application is not prevented from implementing these commands using using NetX and POP3 API services.

5. The NetX POP3 Client API does not provide mail spooling or mail user agent services. A default mail spooler callback is provided as a stub, but the application must define its own mail spooling function to actually transfer mail data to the Client hard drive.

6. NetX POP3 Client is not mail browser ("mail user") for viewing mail data. It is a "mail transfer agent" only. Received mail messages must be properly formatted with the necessary header fields and message body for display by a mail user (browser) agent.

7. The NetX POP3 Server API does not provide a direct means to access mail message data from a hard drive or non volatile storage device on the POP3 Server. The POP3 Server application must define the callbacks specified in **NetX POP3 Server Callbacks** to actually obtain maildrop and mail message data.

8. There is no policy or 'archive' service on the NetX POP3 Server for removing mail that has been downloaded to the POP3 Client but left undeleted in the Client maildrop.

NetX POP3 Authentication

A NetX POP3 Client must authenticate itself to a POP3 Server to access a maildrop. It can do so either by using the USER/PASS commands and providing a username and password known to the POP3 Server, or by using the APOP command and MD5 digest described below. The username is typically a fully qualified domain name (contains a local-part and a domain name, separated by an '@' character). When using the POP3 commands USER and PASS, the Client is sending its username and password unencrypted over the Internet. For this reason, the NetX POP3 Client can be configured use/not use USER/PASS authentication by setting the *enable_user_pass_authentication* parameter in the *_nx_pop3_client_create* service.

The NetX POP3 Client can also be configured to use APOP authentication by setting the *enable_APOP_authentication* parameter in the

_nx_pop3_client_create service. The Client has a shared secret known to the POP3 Server (which can be the same as its password). When the Client sends the APOP command, it takes as its only argument an MD5 digest containing the server domain, local time and process ID extracted from the Server greeting, plus the shared secret. The POP3 Server will create an MD5 digest containing the same information and if its MD5 digest matches the Client's, the Client is authenticated.

The NetX POP3 Client will attempt APOP authentication first, and if that fails, it will attempt USER/PASS authentication if *enable_user_pass_authentication* is set. The NetX POP3 Server allows the Client an unlimited number of attempts at APOP and USER/PASS authentication.

The NetX POP3 Client and Server API services do not have the capability to encrypt exchanges between each other, nor mail message data downloaded to the Client.

The POP3 Client Maildrop

Client mail is stored on a POP3 Server in a "maildrop". A Client maildrop on a POP3 Server is represented as a 1 based list of mail items. That is, each mail is referred to by its index in the maildrop list with the first mail item at index 1 (not zero). POP3 commands refer to specific mail items by their index in this list.

The POP3 Protocol State Machine

The POP3 protocol requires that both Client and Server maintain the state of the POP3 session. First, the Client attempts to connect to the POP3 Server. If successful it enters into the POP3 protocol which has three distinct states defined by RFC 1939. The initial state is the Authorization state in which it must identify itself to the Server. In the Authorization state, the POP3 Client can only issue the USER and the PASS commands, and in that order, or the APOP command. See the **POP3 Protocol Commands** section located. The Client can issue the USER again if the Server has already rejected the Client PASS or APOP command.

Once the POP3 Client is authenticated, the Client session enters the Transaction state. In this state, the Client can download and request mail deletion. The commands allowed in the Transaction state are LIST, STAT, RETR, DELE and RSET in any order and as many times as the Client requires. Server marks the specified mail items for deletion but will not delete any mail until the POP3 session enters the Update state.

Once the Client issues the QUIT command, the POP3 session enters the Update state in which it initiates the TCP disconnect from the Server. If the session was in the Transaction state, the Server will then delete all mail items marked for deletion. The Server should not terminate a POP3 session with a Client even if it rejects the Client commands or has no such Client maildrop.

POP3 Client Commands

NetX POP3 protocol uses the following commands after the Client and Server have successfully connected. Client commands and one line Server replies must be terminated by 0x0D 0x0A.

Command Meaning

The following commands are only allowed in the Authorization state:

USER The Client sends its unencrypted username as the only argument in this command.

PASS The Client sends its unencrypted password as the only argument to the Server to complete authentication with the Server. This command is only allowed after the USER command.

APOP The Client sends an MD5 digest of the Server domain, process ID, and local time as well as the Client's shared secret as the only argument in this command.

The following commands are only allowed in the Transaction state:

STAT This command has no arguments and requests the number of mail items in the Client maildrop and the total bytes of mail item data.

LIST If this command is sent with no arguments, the Server replies with a first line containing the number of mail items in the Client maildrop and the total bytes of mail item data, followed by a line by line listing of each mail item and its message data size.

To receive data for a specific mail item, the Client sends the List command with a single numeric argument corresponding to the mail item index.

RETR To download a specific mail item, the Client sends this command with the mail item index. The mail item will remain in the maildrop regardless if the download succeeds or not.

Typically the Server reply containing “+OK” and the actual mail item data are sent in separate packets, but not always. The Client receives packet data from the Server until it detects an end-of-message sequence (see above) in the Server packet data.

DELE To delete a mail item in its maildrop, the Client sends this command with the mail item index to the Server. The mail item is marked for deletions, and is not actually deleted until the Client terminates the session.

RSET This command takes no arguments and requests the Server to unmark all mail items in the Client maildrop marked for deletion. This command is only allowed in the Transaction state.

QUIT The Client uses this command to terminate the POP3 session at any time.

NOOP This command takes no arguments. The Client can use this command to receive an OK-acknowledgement from the Server.

The syntax for POP3 commands and command parameters is fairly simple. Client commands are case insensitive. The NetX POP3 Client API uses only uppercase for Client command handlers.

Each command is followed either by a carriage return – line feed (CRLF or in hex 0x0D 0x0A) if the command has no arguments, or by a single space and the command argument, followed by the CRLF.

POP3 Server Reply Codes

+OK The Server uses this reply to accept a Client command. There Server can include additional information after the ‘+OK’ but cannot assume the Client will process this information, except in the case of downloading mail message data or the LIST or STAT commands.

-ERR The Server uses this reply to reject a Client command. The Server may send additional information following the ‘-ERR’ but cannot assume the Client will process this information.

The Server reply syntax is similar to the Client command syntax, except Server replies are case sensitive. If there is additional text or reply parameters, the +OK/-ERR is followed by a single space then the text or parameter. Additional parameters are single spaced thereafter. All replies are CRLF terminated.

The exceptions are the mail message download and the 'LIST x' reply. In the mail message download, the Server terminates the mail message data with the end of message sequence (0x0D 0x0A 0x2E 0x0D 0x0A or <CFLF><.><CRLF>) so the Client knows when it has received all the data. For the 'LIST x' reply, the Server ends each line of its reply with a CRLF and ends the last line with the end of message sequence.

The POP3 Protocol Sequence

The general sequence of POP3 protocol is as follows:

1. The Client connects with the Server by way of greeting and initiating a POP3 session.

2. The Server responds to the Client greeting with a "+OK" reply and optional text.

3. The Client sends either the USER or APOP command and waits for Server acknowledgment.

4. If the Server received the APOP command, it attempts to authenticate the Client and if successful, it responds with another "+OK". If the Server received the USER command, it responds with a "+OK" and expects the PASS command next.

5. If the Client sent the USER command, it must send the PASS command next. If it sent the APOP command and the Server rejected it, it may try USER/PASS authentication and send the USER command next. If the Server accepted its APOP command and authentication, the Client may proceed to Step #7.

6. The Server responds to the PASS command with another "+OK" if the Client username and password passes its authentication check. Then the POP3 session enters the Transaction state. If however, the Server rejects the Client password, it sends an "-ERR". The session remains in the Authorization state and the Client can quit or retry sending the APOP or USER/PASS commands.

7. The authenticated Client then queries the Server for how much mail is in its maildrop. It can do this using the LIST or the STAT command. If the Server replies with a "+OK" if it accepts the Client command and is able to access to the maildrop.

8. The Client then issues a RETR command followed by the mail item number for each mail item it wants to download (which is usually all of them). If the Server accepts the command, the Server reply contains a first line beginning with the +OK, followed by the mail message data.

9. The Client should then spool downloaded mail to hard disk to free up its memory resources for the next mail item to download.

10. The Client should next send the DELE command with a specific mail item index to remove mail successfully downloaded from the Server, although it is not required.

    POP3 Servers may have policies regarding how long mail that has been retrieved can remain in the maildrop in order to free up its storage space. So the Client should not assume the Server will save mail left undeleted in its maildrop indefinitely.

11. The Client keeps repeats the RETR command and optionally the DELE command until it has downloaded all its mail (or as much as it desires to download).

12. The Client then terminates the session with the QUIT command and disconnects from the Server. The Server acknowledges with an +OK, enters the Update state, and deletes all maildrop items marked for deletion.

Sample POP3 Client - Server Sessions

Basic POP3 example using USER/PASS:

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: USER mrose
S: +OK mrose is valid
C: PASS mvan99
S: +OK mrose is logged in
C: STAT
S: +OK 2 320
C: RETR 1
```

```
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

Basic POP3 example using APOP (and LIST instead of STAT):

```
S: <wait for connection on TCP port 110>
C: <open connection>
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK mrose's maildrop has 2 messages (320 octets)
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <the POP3 server sends message 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <the POP3 server sends message 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

NetX POP3 Client - Server Sessions

Both NetX POP3 Client and Server APIs provide a ‘session run’ service, *nx_pop3_session_run* and *nx_pop3_server_session_run*, respectively, for connecting a POP3 Client to a POP3 Server and executing the POP3 protocol. The **Small Example System** section demonstrates the use of the NetX POP3 Client and Server using their ‘session run’ services.
In the session run service, the Client and Server exchange commands and replies one to one. If this one to one exchange is disrupted, the

Client and Server may end up waiting to receive the other's next transmission and appear to hang up until their respective session eventually times out.

Below is a brief description of the Client and Server session run service:

The NetX POP3 Client *nx_pop3_session_run* is responsible for getting the Client authenticated, determining how much mail is in the Client maildrop, retrieving the mail, and requesting the Server to delete the mail before quitting and disconnecting from the Server. The *nx_pop3_session_run* has an index field, *maildrop_index,* for keeping track of which item in the maildrop is being acted on. To go through an entire Client maildrop, the *nx_pop3_session_run* increments the *maildrop_index* between successive RETR/DELE calls, from the first to the last maildrop item. POP3 Client applications not using *nx_pop3_session_run* must set the session *maildrop_index* field for each call to RETR, DELE and if desiring a specific mail item, LIST.

- If *nx_pop3_ session_run* encounters an error e.g. with memory allocation or socket transmission, it will return a non zero error status which will abort the current POP3 session, and attempt to inform the Server. If a 'non fatal' error such as a Client command is rejected by the Server or the Client is unable to spool mail to hard drive, it will log an error but allow the session to continue.

- After each Client POP3 session run, the NetX POP3 Client MUST reinitialize the session before connecting with another POP3 Server. This generally involves reinitializing session parameters and disconnecting the current the session socket connection.

The *nx_pop3_server_session_run* is responsible for accepting Client connections, authenticating the Client and servicing its requests till the Client quits session, at which point the *nx_pop3_server_session_run* service will delete any mail marked for deletion. The POP3 Server API does not provide any services for creating the actual Client maildrops or receiving mail for the Client and storing to Client maildrop.

- If *nx_pop3_server_session_run* encounters an error e.g. with memory allocation or socket transmission, it will return a non-zero error status which will abort the current POP3 session, and attempt to inform the Client. If a 'non fatal' error such as a Client command is rejected by the Server or the Server is unable to access the Client maildrop, it will log an error but allow the session to continue.

- After each Server POP3 session run, the NetX POP3 Server MUST reinitialize the session before accepting another Client connection. Similar to the NetX Client, this generally involves reinitializing session parameters and disconnecting the current the session socket connection.

Both NetX POP3 Client and Server APIs provide enough services for the host application to implement its own 'session run' service using the POP3 Client and Server API command and reply handler services directly. This is useful for example if a particur POP3 server or client does not strictly adhere to POP3 protocol.

The TCP/IP layer activities such as packet acknowledgement, detecting dropped packets, and IP packet chaining are not visible to the NetX POP3 protocol.

NetX POP3 API Callbacks

Both NetX POP3 Client and Server APIs define callback services which can only be defined by the host application. In some cases, the callback is required, while in others the API will supply a default service or simply skip the service.

NetX POP3 Client Callbacks

The NetX POP3 Client API has only one callback which is optional.

NetX POP3 Client Mail Spooler Callback

The POP3 Client has only one callback, the mail spooler callback. This service is the means for the NetX POP3 Client API to transfer Client mail message data from volatile (temporary) storage to hard disk or permanent storage before the Client deletes the mail instance from the current session.

The callback routine requires a pointer to the mail instance to spool and returns a status indicator if the mail was successfully spooled. The default mail spooler in the demo source file simply displays the mail contents to screen but does not store the mail data anywhere. Eventually all session mail instances are deleted when the session concludes the mail transaction and is reinitialized to subsequent POP3 transactions or when the session itself is deleted.

An application using the NetX POP3 API can take advantage of the ExpressLogic FileX file system package to implement this callback service.

The format of the mail spooler callback routine is very simple and is defined below:

```
UINT client_mail_spooler(NX_POP3_CLIENT_MAIL *mail_ptr);
```

The input parameter is defined as follows:

Parameter	Meaning
mail_ptr	Pointer to the POP3 Client mail instance to spool to permanent storage

NetX POP3 Server Callbacks

Optional Callbacks

```
UINT (*get_clock_time)(CHAR *clock_time);
UINT (*get_process_ID)(CHAR *process_ID);
UINT (*authentication_check)(NX_POP3_SERVER_SESSION *session_ptr,
                             CHAR *username, CHAR *password, UINT *result);
```

NetX POP3 Server Get Clock Time Callback

The POP3 Server get clock time callback gets the host clock time as follows:

04-01-2007 0600 hrs 0 msecs is represented as ‘200704010600000’

This data is inserted into the POP3 Server greeting to the POP3 Client for the Client to extract and build an MD5 digest and use the APOP command to authenticate itelf to the Server. If the host application does not supply this callback service, the NetX POP3 Server *_nx_pop3_server_get_time* simply returns the NX_POP3_SERVER_DEFAULT_TIME, a user configurable setting.

The callback receives a pointer to the buffer to store the clock time. It is the responsibility of the caller to provide the buffer for the clock time. The callback is constrained to limit the clock time string to the length specified by the user configurable setting, NX_POP3_MAX_CLOCK_TIME.

If the callback completes successfully, the callback returns NX_SUCCESS.

The format of the get clock time callback routine is very simple and is defined below:

```
UINT server_get_clock_time(CHAR *clock_time)
```

The input parameter is defined as follows:

Parameter	Meaning
clock_time	Pointer to the buffer to write the clock time string

NetX POP3 Server Get Process ID Callback

The POP3 Server get process ID callback gets the host process ID which is simply a number which combined with the clock time should produce a unique string.

This process ID along with the clock time is inserted into the POP3 Server greeting to the POP3 Client for the Client to extract and build an MD5 digest and use the APOP command to authenticate itelf to the Server. If the host application does not supply this callback service, the NetX POP3 Server *_nx_pop3_server_get_PID* simply returns the NX_POP3_SERVER_DEFAULT_PROCESS_ID, a user configurable setting.

The callback receives a pointer to the buffer to store the clock time. It is the responsibility of the caller to provide the buffer for the clock time. The callback is constrained to limit the clock time string to the length specified by the user configurable setting, NX_POP3_MAX_CLOCK_TIME.

If the callback completes successfully, the callback returns NX_SUCCESS.

The format of the get clock time callback routine is very simple and is defined below:

```
UINT server_get_process_PID(NX_POP3_SERVER_SESSION *session_ptr, CHAR *process_ID)
```

The input parameter is defined as follows:

Parameter	Meaning
Process_ID	Pointer to the buffer to write the process ID string

NetX POP3 Server Authentication Check Callback

The POP3 Server authentication check callback compares the POP3 Client username and password extracted from the Client USER and PASS commands and matches them against a database or list of Client maildrops on its hard drive. If there is a match, the Client is authenticated.

The callback receives a session pointer to access the POP3 Server session list of Client maildrops. The callback receives a pointer to the Client username and Client password and must find a match among the list of Client maildrops. There fourth parameter is a pointer to an indicator parameter to store the result of the authentication check.

If the search completes successfully, regardless if the authentication check succeeds, the callback returns NX_SUCCESS.

The format of the authentication check callback routine is very simple and is defined below:

```
UINT  server_authentication_check(NX_POP3_SERVER_SESSION *session_ptr,
                                  CHAR *username_ptr, CHAR *password_ptr,
                                  UINT *authenticated)
```

The input parameters are defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session
username_ptr	Pointer to Client username
password_ptr	Pointer to Client password
authenticated	Pointer to the authentication check status

Required Callbacks

```
UINT (*create_client_maildrop_list)(NX_POP3_SERVER *server_ptr),
UINT (*get_client_maildrop)(NX_POP3_SERVER_SESSION *session_ptr, CHAR *username);
UINT (*get_client_mailitem_data)(NX_POP3_SERVER_SESSION *session_ptr,
                                 UINT maildrop_index, ULONG *mailitem_bytes);
UINT (*get_mail_message_buffer)(NX_POP3_SERVER_SESSION *session_ptr, UINT mailitem_index,
                                CHAR **buffer_ptr, UINT *bytes_extracted,
                                UINT *bytes_remaining);
UINT (*delete_mail_on_file)(NX_POP3_SERVER_SESSION *session_ptr);
```

NetX POP3 Server Create Client Maildrop List Callback

The POP3 Server create Client maildrop list callback creates a list of Client maildrops on the POP3 Server and loads data (username, password, shared secret, number of mail items, and total number of bytes) for each maildrop in the POP3 Server *client_maildrops*[NX_POP3_SERVER_MAX_MAILDROP_COUNT] array. This callback is invoked as soon as the POP3 Server session receives a valid APOP command or USER/PASS commands and must be able to validate the Client as well as load information about the Client maildrop in the current session.

If the create Client maildrop list completes successfully, the callback returns NX_SUCCESS.

The format of the create Client maildrop list callback routine is very simple and is defined below:

```
UINT server_create_client_maildrop_list(NX_POP3_SERVER_SESSION *session_ptr);
```

The input parameter is defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session

NetX POP3 Server Get Client Maildrop Callback

The POP3 Server get Client maildrop callback takes an input parameter specifiying the Client by username, and searches among its list of Client maildrops for a match. If it does find a match, it sets the maildrop_found parameter to true, and updates the POP3 Server session maildrop with the Client maildrop information (username, password, shared secret, number of mail items, and total number of bytes).

If the search completes successfully, regardless if a match is found, the callback returns NX_SUCCESS. If there is an error accessing the Server maildrops, it returns a non zero status return.

The format of the get Client maildrop callback routine is very simple and is defined below:

```
UINT server_get_client_maildrop(NX_POP3_SERVER_SESSION *session_ptr,
                                CHAR *username_ptr, UINT *maildrop_found);
```

The input parameters are defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session
username_ptr	Pointer to Client username
maildrop_found	Pointer to result of the maildrop search

NetX POP3 Server Get Client Maildrop Data Callback

The POP3 Server get Client maildrop data callback retrieves the number of bytes of a specific maildrop item for the POP3 Server session Client. Its input parameters include the POP3 Server session pointer, and a maildrop index indicating which maildrop to access. There is also an input pointer to number of bytes of the size of the mail item and another input pointer to indicate to the caller if the mail item was found.

To find the specific maildrop, the callback can get the Client username from the session pointer. The callback then uses the *maildrop_index* input to obtain the specific mail item. If there is no maildrop corresponding to the username and index, the callback sets the *mailitem_found* to NX_FALSE.

If the search completes successfully, regardless if the specified mail item is found, the callback returns NX_SUCCESS. If there is an error accessing the Server maildrop, it returns a non zero status return.

The format of the get Client maildrop callback routine is very simple and is defined below:

```
UINT (*get_client_mailitem_data)(NX_POP3_SERVER_SESSION *session_ptr,
                                 UINT maildrop_index, ULONG *mailitem_bytes,
                                 UINT *mailitem_found);
```

The input parameters are defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session
maildrop_index	Index into POP3 Server list of maildrops
mailitem_bytes	Pointer to the total bytes of mail message data in specified maildrop
mailitem_found	Pointer to the indicator that the mail item was found

NetX POP3 Server Get Mail Message Data Callback

The POP3 Server get mail message data callback retrieves part or all of the specified maildrop mail item. The *_nx_pop3_reply_to_retr* service appends this data to the packet buffer of a packet allocated from the POP3 Server packet pool, and transmits it to the Client. The callback can use the packet payload, NX_POP3_SERVER_PACKET_SIZE, as the maximum amount of data that can be retrieved per call. With each data upload, the callback sets the *buffer_ptr* parameter with the location of the (next) chunk of mail message data for the caller, and updates the *bytes_extracted* pointer with the actual number of bytes retrieved, and *bytes_remaining* pointer with the amount of data remaining to retrieve.

Mail message data can be located in any memory buffer accessible to the Server. The demo file allocates a static buffer and simply copies in text. More typically mail message data is stored in a file and can be extracted with FileX or some other file handling service.

To upload the entire mail message in one call, the host application will need to enable fragmentation (*_nx_ip_fragment_enable*) to handle mail messages larger than the POP3 Server packet payload.

To upload the mail message over multiple calls, the callback must keep an internal index of how many bytes have been uploaded or else a pointer into the mail message data which has not been uploaded yet. FileX has such an interal pointer. In this manner, the caller can iteratively call this callback till the entire mail message is uploaded e.g. bytes_remaining is zero.

To find the specific maildrop, the callback can get the Client username from the session pointer. The callback then uses the *maildrop_index* input to obtain the specific mail item. If there is no maildrop corresponding to the username and index, the callback sets the *mailitem_found* to NX_FALSE.

If the search completes successfully, regardless if the specified mail item is found, the callback returns NX_SUCCESS.

The format of the get mail message data callback routine is very simple and is defined below:

```
UINT (*get_mail_message_buffer)(NX_POP3_SERVER_SESSION *session_ptr, UINT mailitem_index,
                                CHAR **buffer_ptr, UINT *bytes_extracted,
                                UINT *bytes_remaining);
```

The input parameters are defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session
mailitem_index	Index into POP3 Server list of maildrops
buffer_ptr	Pointer to the location in memory where the mail message data is located
bytes_extracted	Pointer to how much data was actually extracted
bytes_remaining	Pointer to how much data in the mail item remains to be retrieved

NetX POP3 Server Delete Mail On File Data Callback

The POP3 Server delete mail on file callback deletes all mail marked for deletion from the session Client maildrop on the POP3 Server. To find the specific maildrop, the callback can get the Client username from the session pointer.

If the search completes successfully, regardless if the specified mail item is found, the callback returns NX_SUCCESS.

The format of the delete mail on file callback routine is very simple and is defined below:

```
UINT (*delete_mail_on_file)(NX_POP3_SERVER_SESSION *session_ptr);
```

The input parameters are defined as follows:

Parameter	Meaning
session_ptr	Pointer to POP3 Server session

NetX POP3 Client Memory Allocation Option

The POP3 Client API must store mail downloaded from the Server in temporary storage before the data can be spooled to a permanent storage device. The NetX POP3 Client has two options for storing mail data.

The first option is to allow the application to set up its own memory (e.g. a fixed size buffer in stack memory) to store mail data. This is well suited to a POP3 client application expecting mail message data limited to a known size. To use the static memory option, NX_POP3_CLIENT_DYNAMIC_MEMORY_ALLOC in *nx_pop3_client.h* must not be defined.

The NX_POP3_CLIENT_MAIL struct has a *mail_buffer_ptr* field which must be set to the static buffer, and *mail_buffer_length* field which must be set to the size of the static buffer. The NetX POP3 Client will copy downloaded mail message data to this buffer location up to the mail size indicated for each mail item downloaded from the Server. It re-uses this buffer between RETR commands, so the Client must spool each downloaded mail item to hard disk before downloading another mail item from the Server.

The second option is to use dynamic memory allocation from the NetX POP3 Client byte and block pool. This option is well suited for the application who cannot anticipate the size of its mail message data or whose mail items vary considerably in size from one to the next. To use the dynamic memory option, NX_POP3_CLIENT_DYNAMIC_MEMORY_ALLOC in *nx_pop3_client.h* must be defined.

If NX_POP3_CLIENT_DYNAMIC_MEMORY_ALLOC is defined, the POP3 Client application must set up one byte and one block pool when it creates the POP3 Client instance. The byte pool is used for small memory allocation (e.g. creating the NX_POP3_CLIENT_MAIL instance) while the block pool is used for storing mail message data which typically requires larger chunks of memory.

Each mail item requires the creation of an NX_POP3_CLIENT_MAIL instance. To store actual message data requires the creation of at least one NX_POP3_MESSAGE_SEGMENT instance associated with the NX_POP3_CLIENT_MAIL instance. If the entire mail message data cannot fit into a single NX_POP3_MESSAGE_SEGMENT instance, the NetX POP3 Client creates more NX_POP3_MESSAGE_SEGMENT instances and connects them in a simple linked list till the message data is completely extracted from the Server packets.

The NetX POP3 Client *nx_pop3_session_run* service frees this memory as soon as it is not needed (e.g. after spooling mail to hard disk), to avoid running out of memory, especially block pool memory. This is not required, since the NetX POP3 Client can maintain download and store multiple mail items in memory before spooling to hard disk, but care must be taken not to run out of dynamic memory. The NetX POP3 Client has an *nx_pop3_utility_print_client_reserves* service which lists the total and available amount of memory in the Client reserves.

The **Small Example System** located elsewhere in this document does not use the Client memory allocation option. For an example of POP3 Client and Server applications which use dynamic memory allocation, see the demo files in the Examples folder.

NetX POP3 Event Logging

The NetX POP3 Client API includes an event logging service **NX_POP3_CLIENT_EVENT_LOG** to log events and data during a POP3 Client operation. The NetX POP3 Server API includes a similar logging service **NX_POP3_SERVER_EVENT_LOG**.

The NetX POP3 Client and Server event logging macros are defined in *nx_pop3_client.h* and *nx_pop3_server.h* respectively. Both use the *printf* statement for displaying output. However, the application code can define its own *printf*.

The Client and Server event logging level is set by NX_POP3_CLIENT_DEBUG in *nx_pop3_client.h* and NX_POP3_SERVER_DEBUG in *nx_pop3_server.h* respectively.

There are four levels of event logging, defined in *nx_pop3.h*, ranging from NONE to ALL.

- NONE: no messages are logged

- LOG: This is intended for the POP3 host application to log specific events during a POP3 session or host application operation. The NetX POP3 Client and Server API do not use this level.

- SEVERE: only events of 'severe' consequences are logged. This would include memory allocation failure, failure to allocate packets, or failed TCP transmission. The POP3 session is not able to recover from these events and the session must abort.

- MODERATE: events of moderate or severe level are logged. For the Server, moderate significance includes packet operations suchs as appending data, accessing Client maildrops, or internal operations. For the Client, this includes commands rejected by the Server, failed authentication attempts, and internal processing errors. Usually the session is able to recover from these events, even if the mail transaction fails.

- ALL: all events are logged. These events include successful mail transaction, successful authentication, and successful session initialization and completion. This level of event logging is intended for testing and debugging the POP3 application. Normal mail traffic would produce too much data to be practically useful.

Below is an example NX_POP3_CLIENT_EVENT_LOG logging call (the NetX POP3 Server event logging is identically formatted):

```
/* Log the event. */
NX_POP3_CLIENT_EVENT_LOG(MODERATE, ("Session %d may not issue a
                          PASS command in the ""%s state. \r\n",
                          session_ptr ->session_id,
                          state_name));
```

The message includes the session ID in which the event occurs, and the error status, and may include additional identifying information.

NetX POP3 Multi-Session Support

A NetX POP3 Client application can be configured to have multiple POP3 sessions running simultaneously. The application code is responsible for creating a single NetX POP3 Client instance and one or more Client sessions. Each Client session requires a session thread entry function defining how it connects with a POP3 Server and conducts the POP3

session. After each POP3 session with a Server, the Client session must be reinitialized before attempting to connect with a POP3 Server again.

The number of Client sessions created must be defined for the NetX POP3 Client API using the user configurable NX_POP3_CLIENT_SESSION_COUNT parameter. The default NetX POP3 Client session count is one. There is no limit other than system resources on the number of session threads a POP3 Client application can create.

A NetX POP3 Server application can also be configured to have multiple POP3 sessions running simultaneously. The application code is responsible for creating a single NetX POP3 Server instance and one or more Server sessions. Each Server session requires a session thread entry function defining how it connects with a POP3 Server and conducts the POP3 session. However, unlike the NetX POP3 Client, the POP3 Server session thread entry function is defined by the NetX POP3 Server API in *nx_pop3_server_session_thread_entry*, not the application code.

Another difference with the NetX POP3 Client is that the POP3 Server API defines the Server thread entry function, *_nx_pop3_server_thread_entry*. The Server thread initializes all its sessions and session sockets. One Server session is chosen to listen for and accept the request while the Server sets up the next available session to listen for the next Client request.

After each POP3 session with a Client, the Server session must be reinitialized before attempting to accept another POP3 Client connection.

The installation **NetX POP3 Examples** folders contains more advanced sample POP3 Client and POP3 Server applications running multiple sessions.

RFCs Supported by NetX POP3

NetX POP3 API is compliant with 1939.

Chapter 2 Installation and Use of NetX POP3

This chapter contains a description of various issues related to installation, setup, and usage of the NetX POP3 Client component.

NetX POP3 Installation

NetX POP3 Client is shipped on a single CD-ROM compatible disk. The package includes three source files, two include files, and a PDF file that contains this document, as follows:

nx_pop3_client.c C Source file for NetX POP3 Client API
nx_pop3_client.h C Header file for NetX POP3 Client API
nx_pop3_server.c C Source file for NetX POP3 Server API
nx_pop3_server.h C Header file for NetX POP3 Server API
nx_pop3.h C Header file for services and definitions common to NetX POP3 Server and Client
nx_md5.c C Source file defining MD5 digest services.
nx_md5.h C Header file defining MD5 digest services.

nx_pop3.pdf PDF description of NetX POP3 API for Client and Server applications

To use NetX POP3 Client API, the entire distribution mentioned previously should be copied to the same directory where NetX is installed. For example, if NetX is installed in the directory "*threadx**mcf5272**green*" then the *nx_md5.h*, *nx_pop3.h*, *nx_pop3_client.h,* and *nx_pop3_client.c* files should be copied into this directory. Similarly, using the NetX POP3 Server API requires copying *nx_md5.h*, *nx_pop3.h*, *nx_pop3_server.h,* and *nx_pop3_server.c* files into the same directory as ThreadX and NetX.

Using NetX POP3

Using the NetX POP3 Client API is easy. The application must add *nx_pop3_client.c* to its build project. The application code must include *nx_md5.h, nx_pop3.h* and *nx_pop3_client.h* after *tx_api.h* and *nx_api.h*, in order to use ThreadX and NetX.

Using the NetX POP3 Server API is similar. The application must add *nx_pop3_server.c* in the build project. The application code must include *nx_pop3.h* and *nx_pop3_server.h* after *tx_api.h* and *nx_api.h*, in order to use ThreadX and NetX.

These files must be compiled in the same manner as other application files and the object code must be linked along with the files of the application. This is all that is required to use the NetX POP3 Client and Server API.

Small Example Client Server System

An example of how to use NetX POP3 is described in Figure 1 that appears below. In this example, the option to use the Client dynamic memory pool allocation is **not enabled.** Below, the POP3 Client and Server are created with a single session, and the NetX and other required components are set up in "*tx_application_define*" in line 100. Line numbers refer to the file line number (1st column of numbers), not the function line number (2nd column of numbers).

In line 81 the Server and Client callback services are listed. The POP3 Server is created first in line 157, its session is created on line 187 and the Server thread is allowed to start on line 302. The POP3 Client is created after the Server on line 214. After successful creation, the POP3 Client session thread is then allowed to start on line 308. In *demo_client_session_thread_entry* which begins on line *319*, the POP3 Client session creates a mail item on line 362, and then connects to the Server and enters into a POP3 session by calling *nx_pop3_session_run* on line 371. The POP3 session is then complete and the Client is deleted on line 386.

The POP3 Server and Client callback services are defined thereafter.

```
1        /*
2           demo_netx_pop3.c
3
4           This is a small demo of POP3 on the high-performance NetX TCP/IP stack.
5           This demo relies on Thread, NetX and POP3 Client and Server API to conduct
6           a POP3 mail session.
7         */
8
9
10       #include  <stdio.h>
11       #include  "nx_api.h"
12       #include  "nx_ip.h"
13       #include  "nx_pop3.h"
14       #include  "nx_pop3_server.h"
15
16
17       /* RAM 'network driver' enables POP3server and client to communicate via RAM. */
18       void    _nx_ram_network_driver(struct NX_IP_DRIVER_STRUCT *driver_req);
19
20
21       /* Set up the POP3 Client.  */
22       #include  "nx_pop3_client.h"
23
24       TX_THREAD           demo_client_thread;
25       NX_POP3_CLIENT      demo_client;
26       NX_PACKET_POOL      client_packet_pool;
```

```
27
28      /* Allocate stack memory for storing downloaded mail.  */
29      NX_POP3_CLIENT_MAIL client_mail;
30      CHAR                client_mail_buffer[NX_POP3_CLIENT_MAIL_BUFFER_SIZE];
31
32      /* Set up Client thread entry point. */
33      void    demo_client_session_thread_entry(ULONG info);
34
35      /* Set up user defined mail spooler service. */
36      UINT client_mail_spooler(NX_POP3_CLIENT_MAIL *mail_ptr);
37
38      /* Allocate stack memory for storing downloaded mail since this
39         demo is NOT using Client dynamic memory to do so.  */
40      NX_POP3_CLIENT_MAIL client_mail;
41      CHAR                client_mail_buffer[NX_POP3_CLIENT_MAIL_BUFFER_SIZE];
42
43      /* Create multiple username and password sets so we can get mail for
44         multiple users with maildrops on the POP3 Server. */
45      #define LOCALHOST                               "POP3_client"
46      #define LOCALHOST_PASSWORD                      "secret_pwd"
47
48
49
50      /* Set up the POP3 Client and Server IP addresses. */
51      #define SERVER_IPADR    IP_ADDRESS(1,2,3,4)
52      #define CLIENT_IPADR    IP_ADDRESS(1,2,3,5)
53
54      /* Create an instance of a Client maildrop on the Server*/
55      NX_POP3_SERVER_MAIL client_maildrop_mail;
56
57      /* Allocate stack memory for retrieving mail in Client maildrop.  */
58
59      #define             MAX_SERVER_MAIL_BUFFER                1000
60      CHAR                server_mail_buffer[MAX_SERVER_MAIL_BUFFER];
61
62      TX_THREAD           demo_server_thread;
63      NX_POP3_SERVER      demo_server;
64      NX_PACKET_POOL      server_packet_pool;
65
66
67      /* IP instances for Client and Server. */
68      NX_IP               client_ip;
69      NX_IP               server_ip;
70
71      /* The POP3 Server requires dynamic memory resources for
72         handling Client maildrop mail items.  */
73      TX_BYTE_POOL        server_byte_pool;
74      TX_MUTEX            server_byte_pool_mutex;
75
76      /* Set up Server thread entry point. */
77      void    demo_server_thread_entry(ULONG info);
78
79
80      /* Set up application defined POP3 server callbacks. */
81      UINT server_create_client_maildrop_list(NX_POP3_SERVER *server_ptr);
82      UINT server_get_client_maildrop(NX_POP3_SERVER_SESSION *session_ptr, CHAR *username_ptr,
                                        UINT *maildrop_found);
83      UINT server_get_client_mailitem_data(NX_POP3_SERVER_SESSION *session_ptr,
                                        UINT maildrop_index, ULONG *mailitem_bytes,
                                        UINT *maildrop_found);
84      UINT server_get_mail_message_buffer(NX_POP3_SERVER_SESSION *session_ptr,
                                        UINT mailitem_index, CHAR **buffer_ptr,
                                        UINT *bytes_extracted, UINT *bytes_remaining);
85      UINT server_get_process_ID(CHAR *process_ID);
86      UINT server_get_clock_time(CHAR *clock_time);
87      UINT server_authentication_check(NX_POP3_SERVER_SESSION *session_ptr,
                                        CHAR *username_ptr, CHAR *password_ptr, UINT *authenticated);
88      UINT server_delete_mail_on_file(NX_POP3_SERVER_SESSION *session_ptr);
89
90
91      /* Define main entry point.  */
92      int main()
93  1   {
94  2       /* Enter the ThreadX kernel.  */
95  3       tx_kernel_enter();
96  4   }
97
98
99      /* Define what the initial system looks like.  */
100     void    tx_application_define(void *first_unused_memory)
101 1   {
```

```
102 2
103 3    UINT   status;
104 4    UINT     i;
105 5    UCHAR    *free_memory_pointer;
106 6
107 7
108 8        /* Setup the pointer to unallocated memory.  */
109 9        free_memory_pointer =  (UCHAR *) first_unused_memory;
110 10
111 11       /* Create Server byte pool for small memory allocation. */
112 12       status = tx_byte_pool_create(&server_byte_pool, NX_POP3_SERVER_BYTE_POOL_NAME,
113 13                                    (VOID **)free_memory_pointer, 1024);
114 14
115 15          /* Create Server byte pool mutex for protecting access to the Server byte pool. */
116 16       status = tx_mutex_create(&server_byte_pool_mutex,
117 17                                NX_POP3_SERVER_BYTE_POOL_MUTEX_NAME, TX_NO_INHERIT);
118 18
119 19       /* Update pointer to unallocated (free) memory. */
120 20       free_memory_pointer =  free_memory_pointer + 1024;
121 21
122 22       /* Create Server packet pool. */
123 23       status =  nx_packet_pool_create(&server_packet_pool, "POP3 Server Packet Pool",
124 24                                       1500, free_memory_pointer,
125 25                                       (1500 * 10));
126 26
127 27       /* Update pointer to unallocated (free) memory. */
128 28       free_memory_pointer = free_memory_pointer + NX_POP3_SERVER_PACKET_POOL_SIZE;
129 29
130 30       /* Initialize the NetX system. */
131 31       nx_system_initialize();
132 32
133 33       /* Create IP instance for demo Server */
134 34       status = nx_ip_create(&server_ip, "POP3 IP Instance", SERVER_IPADR, 0xFFFFFF00UL,
135 35                             &server_packet_pool, _nx_ram_network_driver, free_memory_pointer,
136 36                             1024, NX_POP3_SERVER_IP_THREAD_PRIORITY);
137 37
138 38       /* Update pointer to unallocated (free) memory. */
139 39       free_memory_pointer =  free_memory_pointer + 1024;
140 40
141 41       /* Enable ARP and supply ARP cache memory. */
142 42       status =  nx_arp_enable(&server_ip, (void **) free_memory_pointer,
143 43                               NX_POP3_SERVER_ARP_CACHE_SIZE);
144 44
145 45       /* Update pointer to unallocated (free) memory. */
146 46       free_memory_pointer = free_memory_pointer + NX_POP3_SERVER_ARP_CACHE_SIZE;
147 47
148 48       /* Enable TCP for Server IP. */
149 49       status =  nx_tcp_enable(&server_ip);
150 50
151 51
152 52       /* The demo client username and password is the authentication
153 53          data used when the server attempts to authentication the client. */
154 54
155 55
156 56       /* Create the POP3 Server instance. */
157 57       status = nx_pop3_server_create(&demo_server, &server_ip,
158 58                   free_memory_pointer, 1024,
159 59                   NX_POP3_SERVER_THREAD_PRIORITY, NX_POP3_SERVER_PREEMPTION_THRESHOLD,
160 60                   NX_POP3_SERVER_THREAD_TIME_SLICE, TX_DONT_START, &server_packet_pool,
161 61                   &server_byte_pool, &server_byte_pool_mutex,
                         NX_POP3_SERVER_BYTE_POOL_MUTEX_WAIT,
162 62                   server_get_clock_time, server_get_process_ID, server_authentication_check,
163 63                   server_get_client_maildrop, server_get_client_mailitem_data,
164 64                   server_create_client_maildrop_list, server_get_mail_message_buffer,
165 65                   server_delete_mail_on_file);
166 66
167 67       /* Check for error. */
168 68       if (status != NX_SUCCESS)
169 69       {
170 70
171 71           /* Log the event. */
172 72           NX_POP3_SERVER_EVENT_LOG(SEVERE, ("Error creating POP3 Server.
                                                  Status 0x%x.\n\r", status));
173 73
174 74           /* Abort the application. */
175 75           return;
176 76       }
177 77
178 78       /* Update pointer to unallocated (free) memory. */
179 79       free_memory_pointer += 1024;
180 80
```

```
181 81       /* Create the Server session(s).  In the default NetX POP3 Server API, this is
182 82          just one session. */
183 83       for (i = 0; i < NX_POP3_MAX_SERVER_SESSIONS; i++)
184 84       {
185 85
186 86           /* Create a Server session. */
187 87           status = nx_pop3_server_session_create(&demo_server,
                                &(demo_server.nx_pop3_server_session_list[i]),
188 88                                i+1, free_memory_pointer,
189 89                                1024,
190 90                                NX_POP3_SERVER_SESSION_THREAD_PRIORITY,
191 91                                NX_POP3_SERVER_SESSION_THREAD_PRIORITY,
192 92                                NX_POP3_SERVER_SESSION_THREAD_TIME_SLICE, TX_DONT_START);
193 93
194 94           /* Check for error. */
195 95           if (status != NX_SUCCESS)
196 96           {
197 97
198 98               /* Log the event. */
199 99               NX_POP3_SERVER_EVENT_LOG(SEVERE, ("Error creating server session.
                                                      Status 0x%x\r\n", status));
200 100
201 101              /* Abort the application. */
202 102              return;
203 103          }
204 104
205 105          /* Set the Server session as 'available' to accept POP3 Client connection. */
206 106          demo_server.nx_pop3_server_session_list[i].available = NX_TRUE;
207 107
208 108          /* Move pointer to remaining free memory. */
209 109          free_memory_pointer += 1024;
210 110      }
211 111
212 112  //////////* Set up the POP3 Client. *//////////////////
213 113
214 114      /* Create Client packet pool. */
215 115      status =  nx_packet_pool_create(&client_packet_pool, "POP3 Client Packet Pool",
216 116                                      1500, free_memory_pointer, (1500 * 4));
217 117
218 118      /* Update pointer to unallocated (free) memory. */
219 119      free_memory_pointer = free_memory_pointer + (1500 * 4);
220 120
221 121      /* Create the Client thread */
222 122      status = tx_thread_create(&demo_client_thread, "client_thread",
                                demo_client_session_thread_entry,
223 123                           0, free_memory_pointer, 1024,
224 124                           NX_POP3_CLIENT_THREAD_PRIORITY,
                                NX_POP3_CLIENT_PREEMPTION_THRESHOLD,
225 125                           NX_POP3_CLIENT_THREAD_TIME_SLICE, TX_DONT_START);
226 126
227 127
228 128      /* Update pointer to unallocated (free) memory. */
229 129      free_memory_pointer =  free_memory_pointer + 1024;
230 130
231 131
232 132      /* Create IP instance for demo Client */
233 133      status = nx_ip_create(&client_ip, "POP3 Client IP Instance", CLIENT_IPADR, 0xFFFFFF00UL,
234 134                           &client_packet_pool, _nx_ram_network_driver, free_memory_pointer,
235 135                           1024, NX_POP3_CLIENT_IP_THREAD_PRIORITY);
236 136
237 137      /* Update pointer to unallocated (free) memory. */
238 138      free_memory_pointer = free_memory_pointer + 1024;
239 139
240 140      /* Enable ARP and supply ARP cache memory. */
241 141      status =  nx_arp_enable(&client_ip, (void **) free_memory_pointer,
242 142                              NX_POP3_CLIENT_ARP_CACHE_SIZE);
243 143
244 144      /* Update pointer to unallocated (free) memory. */
245 145      free_memory_pointer = free_memory_pointer + NX_POP3_CLIENT_ARP_CACHE_SIZE;
246 146
247 147      /* Enable TCP for Client IP. */
248 148      status =  nx_tcp_enable(&client_ip);
249 149
250 150      /* Create a NetX POP3 Client instance with no byte or block memory pools.
251 151         Note that it uses its password for its APOP shared secret. */
252 152      status =  nx_pop3_client_create(&demo_client, LOCALHOST, LOCALHOST_PASSWORD,
                            LOCALHOST_PASSWORD,
253 153                        NX_TRUE /* if true, enables APOP authentication */,
254 154                        NX_TRUE /* if true, enables USER/PASS command (e.g. if APOP fails) */,
255 155                        &client_ip, &client_packet_pool,
256 156                        NULL, NULL, 0, NULL, NULL, 0, /* No dynamic memory. */
```

```
257 157                     NX_POP3_CLIENT_REPLY_TIMEOUT,
258 158                     NX_POP3_CLIENT_WINDOW_SIZE, client_mail_spooler);
259 159
260 160
261 161     /* Check for error. */
262 162     if (status != NX_SUCCESS)
263 163     {
264 164
265 165         /* Log the event. */
266 166         NX_POP3_CLIENT_EVENT_LOG(SEVERE, ("Error creating Client. Status 0x%x.\n\r",
                                                status));
267 167
268 168         /* Abort. */
269 169         return;
270 170     }
271 171
272 172     /* Create a single Client session thread. */
273 173     status = tx_thread_create(&(demo_client.nx_pop3_client_session_list[0].session_thread),
274 174                               "POP3 session thread", demo_client_session_thread_entry,
275 175                               (ULONG)(&(demo_client.nx_pop3_client_session_list[0])),
276 176                               free_memory_pointer, NX_POP3_CLIENT_SESSION_THREAD_STACK_SIZE,
277 177                               NX_POP3_CLIENT_SESSION_THREAD_PRIORITY,
278 178                               NX_POP3_CLIENT_SESSION_PREEMPTION_THRESHOLD,
279 179                               NX_POP3_CLIENT_SESSION_THREAD_TIME_SLICE, TX_DONT_START);
280 180
281 181     /* Check for error. */
282 182     if (status != TX_SUCCESS)
283 183     {
284 184
285 185         /* Log the event. */
286 186         NX_POP3_CLIENT_EVENT_LOG(SEVERE, ("Error creating Client session. "
287 187                                           "Status 0x%x\r\n", status));
288 188
289 189         /* Abort. */
290 190         return;
291 191     }
292 192
293 193     /* Configure the Client session to be available to connect with a POP3 Server. */
294 194     demo_client.nx_pop3_client_session_list[0].available = NX_TRUE;
295 195
296 196 ///////////* POP3 Client setup is complete.  *////////////////////
297 197
298 198
299 199     /* Start the Server first. */
300 200     NX_POP3_SERVER_EVENT_LOG(ALL, ("Server starting POP3 service...\r\n"));
301 201
302 202     nx_pop3_server_start(&demo_server);
303 203
304 204     /* Now start the Client. */
305 205     NX_POP3_CLIENT_EVENT_LOG(ALL, ("Client starting POP3 session\r\n"));
306 206
307 207     /* Start the Client thread. */
308 208     status = tx_thread_resume(&demo_client_thread);
309 209
310 210     return;
311 211 }
312
313     /* Define the Client session thread entry function.
314
315        Connect with the POP3 Server and attempt to download all mail
316        in the Client maildrop.  Repeat this for an arbitrary number
317        of session runs, or until an error occurs, and then quit. */
318
319     void    demo_client_session_thread_entry(ULONG info)
320 1   {
321 2
322 3   UINT                    status;
323 4   NX_POP3_CLIENT_SESSION  *session_ptr;
324 5
325 6
326 7       /* Display starting memory reserves. We should get all these back at
327 8          the end of the session. */
328 9       NX_POP3_CLIENT_EVENT_LOG(LOG,("Print client memory reserves at the start "
329 10                                    "of the POP3 session...\r\n"));
330 11
331 12      nx_pop3_utility_print_client_reserves(&demo_client);
332 13
333 14      /* Set up a local pointer to the Client session for convenience. */
334 15      session_ptr = &(demo_client.nx_pop3_client_session_list[0]);
335 16
336 17      /* Initialize POP3 session with server ip address, port and association with Client. */
```

```
337 18       status = nx_pop3_session_initialize(session_ptr, 1, &demo_client,
338 19                                           NX_TRUE /* NX_POP3_CLIENT_DELETE_MAIL_ON_SERVER */,
339 20                                           SERVER_IPADR, NX_POP3_SERVER_PORT);
340 21
341 22       /* Check for errors. */
342 23       if (status != NX_SUCCESS)
343 24       {
344 25
345 26           /* Log the event. */
346 27           NX_POP3_CLIENT_EVENT_LOG(SEVERE, ("Error initializing client session. "
347 28                                             "Status: 0x%x.\n\r", status));
348 29
349 30           /* Abort the Client session. */
350 31           return;
351 32       }
352 33
353 34
354 35       /* Clear memory for a Client mail instance. */
355 36       memset(&client_mail, 0, sizeof(NX_POP3_CLIENT_MAIL));
356 37
357 38       /* Associate Client mail instance with the current session. */
358 39       client_mail.session_ptr = session_ptr;
359 40
360 41       /* Add the mail instance into the session linked list of Client mail.
361 42          At this point there is only this one instance in the session 'list'.*/
362 43       status = nx_pop3_mail_add(session_ptr, &client_mail);
363 44
364 45       /* Make this the session's current mail. */
365 46       session_ptr -> current_mail_ptr = &client_mail;
366 47
367 48       /* Set up session mail message storage variables. */
368 49       session_ptr -> current_mail_ptr -> mail_buffer_ptr = &client_mail_buffer[0];
369 50
370 51       /* Connect with a POP3 server and run a POP3 session. */
371 52       status = nx_pop3_session_run(session_ptr);
372 53
373 54
374 55       /* Disply memory and packet pool reserves available.  Since we are not
375 56          using dynamic memory allocation, this will just display packet pool
376 57          status. */
377 58       NX_POP3_CLIENT_EVENT_LOG(LOG,("Print client memory reserves after session run...\r\n"));
378 59
379 60       nx_pop3_utility_print_client_reserves(&demo_client);
380 61
381 62
382 63       /* Delete the POP3 Client.  This will also delete Client session(s) and session mail.
383 64          In this case since there is no Client dynamice memory to release.  However, there
384 65          is the session socket to delete, the session port to release, and the session
385 66          thread itself to terminate. */
386 67       status = nx_pop3_client_delete(&demo_client);
387 68
388 69       /* Log the client delete status. */
389 70       NX_POP3_CLIENT_EVENT_LOG(ALL, ("Client deleted. Status 0x%x\n\r", status));
390 71   }
391
392
393      /* This default 'spooler' simply displays the mail, in lieu of actually
394         writing to storage device, and returns successful completion. A
395         successful return status means the POP3 Client will delete
396         memory resources allocated for the mail instance just spooled. */
397
398      UINT client_mail_spooler(NX_POP3_CLIENT_MAIL *mail_ptr)
399 1    {
400 2
401 3        /* Display mail item. */
402 4        nx_pop3_utility_print_client_mailitem(mail_ptr);
403 5
404 6        /* Return successful spool to hard disk result. */
405 7        return NX_SUCCESS;
406 8    }
407
408
409      /*************************************************/
410      /* Start of demo POP3 Server callback routines. */
411      /*************************************************/
412
413
414      /* This service creates a list of Client maildrops stored on the POP3 Server.
415         This does not load actual mail message data.  It fills in the Server's array
416         of maildrops with with Client username and total amount of mail data.
417
```

```
418         To make changes to the Server list of Client maildrops, stop and (re)start the
419         using the NetX API nx_pop3_server_stop and nx_pop3_server_start services.
420         This will rebuild the maildrop list.  */
421
422       UINT  server_create_client_maildrop_list(NX_POP3_SERVER *server_ptr)
423 1     {
424 2
425 3     NX_POP3_SERVER_MAILDROP *client_maildrop_ptr;
426 4
427 5         /* Note that the NX_POP3_SERVER_MAX_MAILDROP_COUNT must be
428 6            set to the number of Client maildrops created here for
429 7            the NetX POP3 Server API. */
430 8         server_ptr -> client_maildrop_count = 1;
431 9
432 10        /* Fill in Client data. Note that we are using the password for
433 11           the shared secret used in APOP authentication. */
434 12        client_maildrop_ptr = &(server_ptr -> client_maildrops[0]);
435 13        memset(client_maildrop_ptr, 0, sizeof(NX_POP3_SERVER_MAILDROP));
436 14        client_maildrop_ptr -> client_username = LOCALHOST;
437 15        client_maildrop_ptr -> client_password = LOCALHOST_PASSWORD;
438 16        client_maildrop_ptr -> shared_secret = LOCALHOST_PASSWORD;
439 17        client_maildrop_ptr -> total_mail_items = 2;
440 18        client_maildrop_ptr -> total_bytes = 3090;
441 19
442 20
443 21        /* Return successful completion*/
444 22        return NX_SUCCESS;
445 23    }
446
447       /* This service gets the requested Client maildrop by locating the maildrop using Client
448          username, and setting the session current maildrop to this Client maildrop.  The
449          server_create_client_maildrop_list() must be called before this service
450          can be used. */
451
452       UINT  server_get_client_maildrop(NX_POP3_SERVER_SESSION *session_ptr,
453                                        CHAR *username_ptr, UINT *maildrop_found)
454 1     {
455 2
456 3     UINT                    i;
457 4     NX_POP3_SERVER_MAILDROP *client_maildrop_ptr;
458 5     NX_POP3_SERVER_MAILDROP *session_maildrop_ptr;
459 6
460 7         /* Initialize local variables. */
461 8         i = 0;
462 9         *maildrop_found = NX_FALSE;
463 10
464 11        /* Search thru the Server maildrop list. */
465 12        while (i < session_ptr -> server_ptr -> client_maildrop_count)
466 13        {
467 14
468 15            /* Set a local pointer for convenience. */
469 16            client_maildrop_ptr = &(session_ptr -> server_ptr -> client_maildrops[i]);
470 17
471 18            /* Try to match maildrop username and password with the current
472 19               session username and password.  */
473 20            if (!memcmp(username_ptr, client_maildrop_ptr -> client_username,
474 21                        strlen(client_maildrop_ptr -> client_username)))
475 22            {
476 23
477 24                /* Its a match! */
478 25                *maildrop_found = NX_TRUE;
479 26                break;
480 27            }
481 28
482 29            /* Try the next maildrop. */
483 30            i++;
484 31        }
485 32
486 33        /* Was the maildrop found? */
487 34        if (*maildrop_found)
488 35        {
489 36
490 37            /* Yes, set as the session current mail drop. */
491 38            session_maildrop_ptr = &(session_ptr -> client_maildrop);
492 39
493 40            /* Update the Server session maildrop instance with maildrop data. */
494 41            session_maildrop_ptr -> client_username = client_maildrop_ptr -> client_username;
495 42            session_maildrop_ptr -> client_password = client_maildrop_ptr -> client_password;
496 43            session_maildrop_ptr -> shared_secret = client_maildrop_ptr -> shared_secret;
497 44            session_maildrop_ptr -> total_bytes = client_maildrop_ptr -> total_bytes;
498 45            session_maildrop_ptr -> total_mail_items = client_maildrop_ptr -> total_mail_items;
```

```
499 46       }
500 47
501 48       /* Return successful completion */
502 49       return NX_SUCCESS;
503 50   }
504
505
506      /* This service defines the Server authentication check.  This is called by
507         the POP3 server when it receives a Client PASS command.
508
509         A non successful completion status is returned only if a null username/password
510         is received.  The supplied username and password are matched against the
511         list of Server maildrop username/password data till (if) a match is found. */
512
513      UINT  server_authentication_check(NX_POP3_SERVER_SESSION *session_ptr, CHAR *username_ptr,
514                                        CHAR *password_ptr, UINT *authenticated)
515 1    {
516 2
517 3    UINT                    i;
518 4    NX_POP3_SERVER_MAILDROP *maildrop;
519 5
520 6
521 7        /* Check for invalid parameters. */
522 8        if (!username_ptr || !password_ptr || !strlen(username_ptr)  || !strlen(password_ptr))
523 9        {
524 10
525 11           /* Return the error status. */
526 12           return NX_PTR_ERROR;
527 13       }
528 14
529 15       /* Initialize variables. */
530 16       *authenticated = NX_FALSE;
531 17       i =  0;
532 18
533 19       /* Set a local pointer for convenience. */
534 20       maildrop = &(session_ptr -> server_ptr -> client_maildrops[i]);
535 21
536 22       /*Search the Server maildrop list for matching username/password. */
537 23       while (i < session_ptr -> server_ptr -> client_maildrop_count)
538 24       {
539 25
540 26           /* Does the username match this maildrop? */
541 27           if (!memcmp(maildrop -> client_username, username_ptr,
542 28                       strlen(maildrop -> client_username)))
543 29           {
544 30
545 31               /* Yes; now check if the passwords match. */
546 32               if (!memcmp(maildrop -> client_password, password_ptr,
547 33                           strlen(maildrop -> client_password)))
548 34               {
549 35
550 36                   /* Its a match! */
551 37                   *authenticated = NX_TRUE;
552 38               }
553 39
554 40               /* Successful authentication attempt regardless if correct password applied. */
555 41               return NX_SUCCESS;
556 42           }
557 43
558 44           /* Try the next maildrop. */
559 45           i++;
560 46           maildrop = &(session_ptr -> server_ptr -> client_maildrops[i]);
561 47       }
562 48
563 49       /* Return successful completion even though no match was found. */
564 50       return NX_SUCCESS;
565 51   }
566
567
568
569      /* This service retrieves the POP3 server local clock time.  An
570         actual application defined get clock time can access the system
571         device clock while this service uses a static time and date string.
572         Note this service SHOULD check the maximum length the NetX POP3 Server API
573         expects for the clock_time string (NX_POP3_MAX_CLOCK_TIME. */
574
575      UINT server_get_clock_time(CHAR *clock_time)
576 1    {
577 2
578 3        /* Create a dummy time (04-01-2007 0600 hrs 0 msecs. It
579 4           must be less than or equal to the
```

```
580 5           NX_POP3_SERVER_CLOCK_TIME_SIZE. */
581 6       memcpy(clock_time,"200704010600000", strlen("200704010600000"));
582 7       return NX_SUCCESS;
583 8   }
584
585     /* This service retrieves the POP3 server process ID.  An
586        actual application defined get process ID can access the system
587        for this data while this service uses a static ID.
588        Note this service SHOULD check the maximum length the NetX POP3 Server API
589        expects for the clock_time string (NX_POP3_MAX_PROCESS_ID. */
590
591     UINT server_get_process_ID(CHAR *process_ID)
592 1   {
593 2
594 3       /* Create a dummy process ID. It must be less than
595 4          or equal to the NX_POP3_SERVER_PROCESS_ID_SIZE. */
596 5       memcpy(process_ID, "123456789", strlen("123456789"));
597 6       return NX_SUCCESS;
598 7   }
599
600     /*  This service deletes POP3 Server session mail from the storage
601         device on which the Server session maildrop is located. The maildrop
602         is pointed to by the session's current maildrop.  This default service
603         simply prints out a message that the fictitious mail file is
604         deleted.  */
605     UINT server_delete_mail_on_file(NX_POP3_SERVER_SESSION *session_ptr)
606 1   {
607 2
608 3   UINT                    i;
609 4   NX_POP3_SERVER_MAILDROP *client_maildrop_ptr;
610 5   NX_POP3_SERVER_MAIL     *mail_ptr;
611 6   CHAR                    filename[20];
612 7
613 8
614 9       /* Set up local pointers for convenience. */
615 10      client_maildrop_ptr = &(session_ptr -> client_maildrop);
616 11
617 12      /* Release memory used for the current session maildrop. */
618 13      mail_ptr = client_maildrop_ptr -> start_mail_ptr;
619 14
620 15      /* Set up a local variable for each mail message's
621 16         unique 'filename'. */
622 17      i= 1;
623 18
624 19      /* Search all the mail in the client maildrop*/
625 20      while (mail_ptr)
626 21      {
627 22
628 23          /* Is the mail item marked for deletion off the hard drive? */
629 24          if (mail_ptr -> marked_for_deletion == NX_TRUE)
630 25          {
631 26
632 27              /* Yes; Delete the mail off the server hard drive. */
633 28              NX_POP3_SERVER_EVENT_LOG(ALL, ("Deleting next mail file %d.\r\n", i));
634 29
635 30              sprintf(filename, "mail_message_%d.msg", i);
636 31              printf("Deleting file %s\r\n", filename);
637 32          }
638 33
639 34          /*Get the next maildrop item. */
640 35          mail_ptr = mail_ptr -> next_ptr;
641 36
642 37          i++;
643 38      }
644 39
645 40      /* Return successful completion. */
646 41      return NX_SUCCESS;
647 42  }
648
649     /* This service uploads mail message data to the POP3 Server session.
650        For this demo callback, a large buffer created on the stack is used to hold
651        the entire message.  Each message is small enough to be loaded into
652        a single Server packet and transmitted to the Client.  For larger messages
653        exceeding packet payload, the application would have to enable IP
654        packet fragmentation to upload a large message file and send it out
655        in a single TCP socket send call.
656
657        Because Client mail can be any size, this service should be designed
658        to return part of the message data as suits the application needs.
659        This service can take advantage of FileX for example
660        to read chunks of a file at a time for copying to packet data and sending.
```

```
661         See the help document for more details on this callback. */
662
663     UINT server_get_mail_message_buffer(NX_POP3_SERVER_SESSION *session_ptr,
664                                         UINT mailitem_index,
665                                         CHAR **buffer_ptr,
666                                         UINT *bytes_extracted,
667                                         UINT *bytes_remaining)
668 1   {
669 2
670 3       /* Set return values to no data extracted. */
671 4       *bytes_extracted = 0;
672 5
673 6       /* Clear the buffer to hold mail message data. */
674 7       memset(server_mail_buffer, 0, MAX_SERVER_MAIL_BUFFER);
675 8
676 9       /* Fill in the buffer with the specified mail item. */
677 10      switch (mailitem_index)
678 11      {
679 12
680 13          case 1:
681 14
682 15              /* Simulate reading in message data from maildrop file on Server. */
683 16              sprintf(server_mail_buffer,"From: diverjen@netcourrier.com\r\n"
684 17               "Sent: Thursday, April 19, 2007 4:24 AM\r\n"
685 18               "To: jchristiansen@expresslogic.com\r\n"
686 19               "Subject: Message of the Day\r\n"
687 20               "\r\n"
688 21               "Hi Janet\r\n"
689 22               "\r\n"
690 23               "Thanks for thinking of me, but I will be working during that time.\r\n"
691 24               "This year it is a lot more work than play so I will have to decline\r\n"
692 25               "your invitation and stay and work work work so I can pay the bills!\r\n"
693 26               "But good luck and let me know how you do.\r\n"
694 27               "\r\n"
695 28               "I met Nicole this morning at the start of the SD 600k Brevet (she is doing \r\n"
696 29               "the whole thing, I only did the first loop) and she told me about you guys \r\n"
697 30               "have dinner Sunday pm. Can you let me know when and where? She said about \r\n"
698 31               "6:00 pm, but I suppose that depends on when she & Anthony finishes, what \r\n"
699 32               "kind of shape they are in. If it suits my wife (she has somewhat strict \r\n"
700 33               "food requirements) and is not too late I will be bringing her along. \r\n"
701 34               "\r\n"
702 35               "Big Hugs\r\n"
703 36               "\r\n"
704 37               "Jenno\r\n%s", NX_POP3_END_OF_MESSAGE_TAG);
705 38
706 39          break;
707 40          case 2:
708 41              /* Simulate reading in message data from maildrop file on Server. */
709 42              sprintf(server_mail_buffer,"From: cyclesport@worldproductions.com\r\n"
710 43               "Sent: Thursday, May 2, 2007 4:24 PM\r\n"
711 44               "To: jchristiansen@expresslogic.com\r\n"
712 45               "Subject: 2007 Edition of the Tour of Italy (Giro)\r\n"
713 46               "\r\n"
714 47               "Hi Janet\r\n"
715 48               "\r\n"
716 49               "The 90th edition of the Giro d'Italia will be one for the climbers.\r\n"
717 50               "Although starting off in Sardegna with a team time trial, the parcoursr\n"
718 51               "will contain more climbing than time trialing. Parting on May 12,\r\n"
719 52               "the race will make its way from the south to the Alps and then to\r\n"
720 53               "the Dolomites, before finishing in Milano, 3,442 km later, on June 3.\r\n
721 54               "\r\n"
722 55               "'The team time trail will be interesting because it is on the first day, \r\n"
723 56               "and always the riders will still be adjusting,' said Paolo Savoldelli.\r\n"
724 57               "'But the distance is short so it will not be that much of a factor.'"
725 58               "\r\n%s", NX_POP3_END_OF_MESSAGE_TAG);
726 59          break;
727 60
728 61          default:
729 62              NX_POP3_SERVER_EVENT_LOG(MODERATE,("No such mail item in Client maildrop.\r\n"));
730 63
731 64              return NX_POP3_ERROR_BAD_CLIENT_MAILITEM;
732 65      }
733 66
734 67      /* Set the buffer pointer to the mail buffer containing data. */
735 68      *buffer_ptr = server_mail_buffer;
736 69
737 70      /* Indicate this is the entire message (no bytes remaining). */
738 71      *bytes_remaining = 0;
739 72      *bytes_extracted = strlen(server_mail_buffer);
740 73
741 74      /* Return successful completion status. */
```

```
742 75        return NX_SUCCESS;
743 76
744 77    }
745
746       /* This service retrieves the size ('data') of the specified mail item.
747          The server_create_client_maildrop_list() must be called before this service
748          can be used.
749
750          Note that the maildrop index sent in by the caller starts at 1. This is
751          because the syntax for DELE, LIST and RETR commands requires the mail index
752          be specified as a command parameter, and the maildrop list is a 1 based list
753          in POP3 protocol. */
754
755       UINT server_get_client_mailitem_data(NX_POP3_SERVER_SESSION *session_ptr, UINT
                                 maildrop_index, ULONG *mailitem_bytes, UINT *mailitem_found)
757 1     {
758 2
759 3         *mailitem_found = NX_TRUE;
760 4
761 5         if (!memcmp(session_ptr -> client_username, LOCALHOST, strlen(LOCALHOST)))
762 6         {
763 7             switch (maildrop_index)
764 8             {
765 9
766 10                case 1:
767 11                   /* 'Read' the size of mail item in server cache. */
768 12                   *mailitem_bytes = 1790;
769 13                break;
770 14
771 15                case 2:
772 16                   /* 'Read' the size of mail item in server cache. */
773 17                   *mailitem_bytes = 1296;
774 18                break;
775 19
776 20                default:
777 21                   *mailitem_found = NX_FALSE;
778 22            }
779 23        }
780 24        /* Handle a username not found among Server maildrops. */
781 25        else
782 26            *mailitem_found = NX_FALSE;;
783 27
784 28        /* Return successful completion regardless with location of requested mail item. */
785 29        return NX_SUCCESS;
786 30    }
```

Figure 1. Example of POP3 use with NetX

Client Configuration Options

There are several configuration options with the NetX POP3 Client API. Following is a list of all options described in detail:

Define	Meaning
NX_DISABLE_ERROR_CHECKING	Defined, this option removes the basic POP3 error checking. It is typically used after the application has been debugged. The default NetX POP3 Client status is enabled.
NX_POP3_CLIENT_DEBUG	This option sets the level of POP3 Client event logging, from logging ALL messages, to only logging SEVERE errors. To disable logging, set level to NONE. The default NetX POP3 Client level is set to MODERATE.
NX_POP3_CLIENT_DYNAMIC_MEMORY_ALLOC	Defined, this enables the POP3 Client to create and delete mail using the Client byte and block pools for memory allocation and release. The default NetX POP3 Client setting is defined.
NX_POP3_CLIENT_MAIL_BUFFER_SIZE	This defines the size of the Client buffer for storing downloaded mail message data for Clients not configured to use dynamic memory allocation for storing mail data. The Default NetX POP3 setting is 3000 bytes.
NX_POP3_CLIENT_SESSION_COUNT	This option sets the number of the Client sessions with the

POP3 Server(s). The default NetX POP3 Client size is 1.

NX_POP3_CLIENT_THREAD STACK_SIZE

This option sets the size of the Client thread stack. The default NetX POP3 Client size is 4096.

NX_POP3_CLIENT_THREAD_PRIORITY

This option sets the set the Client thread priority. The default NetX POP3 Client value is 2.

NX_POP3_CLIENT_PREEMPTION_THRESHOLD

This option sets the sets the level of priority at which the Client thread allows preemption. The default NetX POP3 Client value is 2.

NX_POP3_CLIENT_THREAD_TIME_SLICE

This option sets the time slice of the scheduler allows for Client thread execution. The default NetX POP3 Client size is TX_NO_TIME_SLICE.

NX_POP3_CLIENT_SESSION_THREAD_STACK_SIZE

This option sets the size of the Client session thread stack. The default NetX POP3 Client size is 4096.

NX_POP3_CLIENT_SESSION_THREAD_TIME_SLICE

This option sets the time slice of the scheduler allows for a Client session thread execution. The default NetX POP3 Client size is TX_NO_TIME_SLICE.

NX_POP3_CLIENT_SESSION_THREAD_PRIORITY

This option sets the Client session thread priority. The default NetX

POP3 Client value is set to NX_POP3_CLIENT_THREAD_PRIORITY.

NX_POP3_CLIENT_SESSION_PREEMPTION_THRESHOLD

This option sets the sets the level of priority at which the Client session thread allows preemption. The default NetX POP3 Client value is 2.

NX_POP3_CLIENT_BYTE_POOL_SIZE

This option sets the NetX POP3 Client byte pool size. The NetX POP3 Client byte pool default size is 2048 bytes.

NX_POP3_CLIENT_BYTE_POOL_ NAME

This option sets the name of the Client byte pool. The NetX POP3 Client default is “Client bytepool.”

NX_POP3_CLIENT_BYTE_POOL_MUTEX_NAME

This option sets the name of the Client byte pool mutex. The NetX POP3 Client byte pool mutex name default is “Client bytepool mutex.”

NX_POP3_CLIENT_BYTE_POOL_MUTEX_WAIT

This option sets the Client byte pool mutex timeout to obtain the mutex. The NetX POP3 Client byte pool mutex timeout is 5 seconds.

NX_POP3_CLIENT_BLOCK_SIZE

This option sets the NetX POP3 Client block pool’s block size. The NetX POP3 Client block pool default size is NX_POP3_CLIENT_PACKET_SIZE.

NX_POP3_CLIENT_BLOCK_POOL_SIZE

This option sets the NetX POP3 Client block pool size. The NetX POP3 Client block pool default size in bytes is 16 * X_POP3_CLIENT_PACKET_SIZE.

NX_POP3_CLIENT_BLOCK_POOL_ NAME

This option sets the name of the Client block pool. The NetX POP3 Client block pool name is default is “Client blockpool.”

NX_POP3_CLIENT_BLOCK_POOL_MUTEX_NAME

This option sets the name of the Client block pool mutex. The NetX POP3 Client block pool mutex name idefault is “Client blockpool mutex.”

NX_POP3_CLIENT_BLOCK_POOL_MUTEX_WAIT

This option sets the Client block pool mutex timeout to obtain the mutex. The default NetX POP3 Client block pool mutex timeout is 5 seconds.

NX_POP3_CLIENT_PACKET_SIZE This sets the size of the TCP packet which carries message data to the POP3 Server. This includes TCP, IP, and Ethernet (Frame) packet header data. The default NetX POP3 Client is 1500.

NX_POP3_CLIENT_PACKET_POOL_SIZE

This option sets the size of the POP3 Client packet pool. The NetX POP3 Client default is (10 * NX_POP3_CLIENT_PACKET_SIZE).

NX_POP3_CLIENT_PACKET_TIMEOUT

This option sets the timeout on NetX packet allocation. The

default NetX POP3 Client packet timeout is 10 seconds.

NX_POP3_TCP_SOCKET_SEND_WAIT

This option sets the timeout on a NetX TCP socket send completion. The default NetX POP3 Client socket send timeout is 2 seconds.

NX_POP3_CLIENT_REPLY_TIMEOUT

This option sets the timeout on a NetX TCP socket receive call for the Server reply to the previous Client command. The default NetX POP3 Client setting is timeout is 10 seconds.

NX_POP3_CLIENT_CONNECTION_TIMEOUT

This option sets the Client TCP socket connection timeout. The default NetX POP3 Client connection timeout is 30 seconds.

NX_POP3_CLIENT_DISCONNECT_TIMEOUT

This option sets the Client TCP socket disconnect timeout. The default NetX POP3 Client connect timeout is 2 seconds.

NX_POP3_CLIENT_TCP_SOCKET_NAME

This option sets the TCP socket name. The NetX POP3 Client TCP socket name default is "POP3 Client socket."

NX_POP3_SERVER_PORT

This option defines the server port for the Client to connect to. The default NetX POP3 Client server port is 110.

NX_POP3_CLIENT_IPADR

This option sets the POP3 Client IP Address. The default NetX POP3 Client is 192.2.2.34.

NX_POP3_CLIENT_IP_THREAD_STACK_SIZE

This option sets the Client IP helper thread stack size. The default NetX POP3 Client size is 1024 bytes.

NX_POP3_CLIENT_IP_THREAD_PRIORITY

This option sets Client IP helper thread priority. The default NetX POP3 Client value is 1.

NX_POP3_CLIENT_ARP_CACHE_MEM_SIZE

This option sets the ARP cache memory size. Each ARP entry is 52 bytes, so the number of ARP entries is the memory size divided by 52. The default NetX POP3 Client ARP cache memory size is 1040 (20 entries).

NX_POP3_CLIENT_WINDOW_SIZE This option sets the size of the Client TCP receive window. This should be set to below the MTU size of the underlying Ethernet hardware. The default NetX POP3 Client TCP window size is NX_POP3_CLIENT_PACKET_SIZE.

.

NX_POP3_MAX_USERNAME This option sets the limit on the size of the buffer containing the POP3 Client user name for connecting to the Server. The default NetX POP3 Client setting is 40 bytes.

NX_POP3_MAX_PASSWORD This option sets the limit on the size of the buffer containing the POP3 Client password for authenticating itself to the Server. The default NetX POP3 Client setting is 20 bytes.

NX_POP3_MAX_SHARED_SECRET This option sets the limit on the size of the buffer containing the POP3 Client shared secret for

	APOP authentication with the Server. The default NetX POP3 Client setting is 20 bytes.
NX_POP3_CLIENT_ENABLE_APOP	This option enables the POP3 Client to use the APOP command to authenticate itself to the Server. The default NetX POP3 Client setting is enabled (NX_TRUE).
NX_POP3_CLIENT_DELETE_MAIL_ON_SERVER	This option if enabled causes the POP3 Client to request mail items successfully downloaded from the POP3 Server to be deleted from the Client maildrop on the Server. The default NetX POP3 Client setting is enabled (NX_TRUE).

Server Configuration Options

There are several configuration settings and options with the NetX POP3 Server API. Following is a list of all options described in detail:

Define	**Meaning**
Configure NetX POP3 Server Debug and Event Logging Parameters	
NX_DISABLE_ERROR_CHECKING	Defined, this option removes the basic POP3 error checking. It is typically used after the application has been debugged. The default NetX POP3 Server status is enabled.
NX_POP3_SERVER_DEBUG	This option sets the level of POP3 Server event logging, from logging ALL messages, to only logging SEVERE errors. To disable logging, set level to NONE. The default NetX POP3 Server level is set to MODERATE.

NX_POP3_PRINT_SERVER_RESERVES

This option enables the print available Server byte pool memory and remaining packets in the Server packet pool service. The default NetX POP3 Server feature is undefined.

Configure NetX POP3 Server and Server Session Thread Parameters

NX_POP3_SERVER_THREAD_STACK_SIZE

This option sets the size of the Server thread on the stack. The default NetX POP3 Server value is 4096.

NX_POP3_SERVER_THREAD PRIORITY

This option sets the set the Server thread priority. The default NetX POP3 Server value is 2.

NX_POP3_SERVER_THREAD_TIME_SLICE

This option sets the time slice of the scheduler allows for Server thread execution. The default NetX POP3 Server value is TX_NO_TIME_SLICE.

NX_POP3_SERVER_PREEMPTION_THRESHOLD

This option sets the sets the level of priority at which the Server thread allows preemption. The default NetX POP3 Server value is NX_POP3_SERVER_THREAD PRIORITY.

NX_POP3_SERVER_SESSION_THREAD_STACK_SIZE

This option sets the size of the Server session thread stack. The default NetX POP3 Server value is 4096.

NX_POP3_SERVER_SESSION_THREAD_TIME_SLICE

This option sets the time slice of the scheduler allows for a Server session thread execution. The default NetX POP3 Server value is TX_NO_TIME_SLICE.

NX_POP3_SERVER_ SESSION_THREAD_PRIORITY

This option sets the Server session thread priority. The default NetX POP3 Server value is NX_POP3_SERVER_THREAD_PRIORITY.

NX_POP3_SERVER_SESSION_PREEMPTION_THRESHOLD

This option sets the sets the level of priority at which the Server session thread allows preemption. The default NetX POP3 Server value is NX_POP3_SERVER_SESSION_THREAD_PRIORITY.

Configure Server memory resources

NX_POP3_SERVER_BYTE_POOL_ NAME

This option sets the name of the Server byte pool. The NetX POP3 Server default is “POP3 Server bytepool.”

NX_POP3_SERVER_BYTE_POOL_SIZE

This option sets the NetX POP3 Server byte pool size. The NetX POP3 Server byte pool default size is 4096 bytes.

NX_POP3_SERVER_BYTE_POOL_MUTEX_NAME

This option sets the name of the Server byte pool mutex. The NetX POP3 Server byte pool mutex name default is “POP3 Server bytepool mutex.”

NX_POP3_SERVER_BYTE_POOL_MUTEX_WAIT

This option sets the Server byte pool mutex timeout to obtain the mutex. The NetX POP3 Server byte pool mutex timeout is 2 seconds.

Configure NetX POP3 Server Network Resources

NX_POP3_SERVER_SESSION_PORT

This option sets the POP3 Server port on which to listen for Client requests. The default NetX POP3 Server value is 110.

NX_POP3_SERVER_SOCKET_QUEUE_SIZE

This option sets the number of Client requests that can be queued in the POP3 Server socket. The default NetX POP3 Server value is 5.

NX_POP3_SERVER_WINDOW_SIZE This option sets the size of the Server TCP receive window. This should be set to below the MTU size of the underlying Ethernet hardware. The default NetX POP3 Server TCP window size is NX_POP3_SERVER_PACKET_SIZE.

NX_POP3_SERVER_PACKET_SIZE This sets the size of the TCP packet which carries message data to the POP3 Client. This includes TCP, IP, and Ethernet (Frame) packet header data. The default NetX POP3 Server is 1500.

NX_POP3_SERVER_PACKET_HEADER_SIZE

This option sets aside the number of bytes of the packet size for header data. The default NetX POP3 Server is 60.

NX_POP3_SERVER_PACKET_POOL_SIZE

This option sets the size of the POP3 Server packet pool. The NetX POP3 Server default is (20 * NX_POP3_SERVER_PACKET_SIZE).

.

NX_POP3_SERVER_PACKET_TIMEOUT

This option sets the timeout on NetX packet allocation. The default NetX POP3 Serverpacket timeout is 1 second.

NX_POP3_SERVER_TCP_SOCKET_SEND_WAIT

This option sets the Server TCP socket send timeout. The default NetX POP3 Server connection timeout is 3 seconds.

NX_POP3_SERVER_IP_THREAD_STACK_SIZE

This option sets the Server IP helper thread stack size. The default NetX POP3 Server size is 2048 bytes.

NX_POP3_SERVER_IP_THREAD_PRIORITY

This option sets Server IP helper thread priority. The default NetX POP3 Server value is 2.

NX_POP3_SERVER_ARP_CACHE_ SIZE

This option sets the ARP cache memory size. Each ARP entry is 52 bytes, so the number of ARP entries is the memory size divided by 52. The default NetX POP3 Server ARP cache memory size is 1040 (20 entries).

NX_POP3_SERVER_TCP_ RECEIVE_TIMEOUT

This option sets the Server TCP socket receive timeout. The default NetX POP3 Server connection timeout is 5 seconds.

NX_POP3_SERVER_CONNECTION_TIMEOUT

This option sets the Server TCP socket connection timeout. The default NetX POP3 Server connection timeout is NX_WAIT_FOREVER (no timeout).

NX_POP3_SERVER_DISCONNECTION_TIMEOUT

This option sets the Server TCP socket disconnect timeout. The default NetX POP3 Server connect timeout is 10 seconds.

Configure NetX POP3 Server Session Parameters

NX_POP3_MAX_SERVER_SESSIONS

This option sets the number of the Server sessions. The default NetX POP3 Server value is 1.

NX_POP3_SERVER_MAX_MAILDROP_COUNT

This option sets the maximum number of POP3 Client maildrops the Server can hold. The default NetX POP3 Server setting is 3.

NX_POP3_SERVER_MAX_REPLY

This option sets Server reply buffer size (so maximum size of the Server reply text to the Client). The default NetX POP3 Server value is 200.

NX_POP3_MAX_CLIENT_USERNAME

This option sets buffer size in the POP3 Server session for storing

Client username. The default NetX POP3 Server value is 40.

NX_POP3_MAX_CLIENT_PASSWORD

This option sets buffer size in the POP3 Server session for storing Client password. The default NetX POP3 Server value is 20.

NX_POP3_MAX_CLIENT_SECRET

This option sets buffer size in the POP3 Server session for storing Client shared secret which is used in the Client MD5 digest for APOP authentication. The default NetX POP3 Server value is 20.

NX_POP3_MAX_SERVER_APOP_STRING

This option sets buffer size in the POP3 Server session for storing text in the Server greeting containing the required APOP data for the Client (e.g. process ID, Server clock time, and Server domain). The default NetX POP3 Server value is 100.

NX_POP3_MAX_CLOCK_TIME This option sets buffer size in the POP3 Server session for storing Server clock time. The default NetX POP3 Server value is 20.

NX_POP3_MAX_PROCESS_ID This option sets buffer size in the POP3 Server session for storing Server session process ID. The default NetX POP3 Server value is 10.

NX_POP3_SERVER_DOMAIN This option sets the POP3 Server domain name which is used in the Server greeting to the POP3 Client. The default NetX POP3 Server value is ‘server.com’.

NX_POP3_SERVER_DEFAULT_TIME This option sets the POP3 Server 'time' in the event the host application does not supply a get clock time callback. The default NetX POP3 Server value is '200706150600000'.

NX_POP3_SERVER_DEFAULT__PROCESS_ID
This option sets the POP3 Server default process ID in the event the host application does not supply a get process ID callback. The default NetX POP3 Server value is '1234'.

NX_POP3_ENABLE_PRINTF Defined, this option enables diagnostic printf calls from the NetX POP3 client and/or server.

Chapter 3 Description of POP3 Client Services

This chapter contains a description of all NetX POP3 Client services (listed below) in alphabetical order.

In the “Return Values” section in the following API descriptions, values in **BOLD** are not affected by the **NX_DISABLE_ERROR_CHECKING** define that is used to disable API error checking, while non-bold values are completely disabled.

nx_pop3_client_connect
Connect POP3 Client Instance to its POP3 Server

nx_pop3_client_create
Create a POP3 Client Instance

nx_pop3_client_delete
Delete a POP3 Client instance

nx_pop3_cmd_dele
Send DELE command to POP3 Server

nx_pop3_cmd_greeting
Send greeting (connect) to POP3 Server

nx_pop3_cmd_list
Send LIST command to POP3 Server

nx_pop3_cmd_noop
Send NOOP command to POP3 Server

nx_pop3_cmd_quit
Send QUIT command to POP3 Server

nx_pop3_cmd_pass
Send PASS command to POP3 Server

nx_pop3_cmd_retr
Send RETRcommand to POP3 Server

nx_pop3_cmd_rset
Send RSET command to POP3 Server

nx_pop3_cmd_stat

Send STAT command to POP3 Server

nx_pop3_cmd_user
Send USER command to POP3 Server

nx_pop3_mail_add
Add mail item to list of session mail downloaded

nx_pop3_mail_create
Add mail item to list of session mail downloaded

nx_pop3_mail_delete
Delete mail item from the list of session mail

nx_pop3_mail_spool
Spool downloaded mail item to device hard disk

nx_pop3_rsp_dele
Handle POP3 Server reply to DELE command

nx_pop3_rsp_greeting
Handle POP3 Server reply to greeting

nx_pop3_rsp_list
Handle POP3 Server reply to LIST command

nx_pop3_rsp_noop
Handle POP3 Server reply to NOOP command

nx_pop3_rsp_quit
Handle POP3 Server reply to QUIT command

nx_pop3_rsp_user
Handle POP3 Server reply to USER command

nx_pop3_rsp_retr
Handle POP3 Server reply to RETR command

nx_pop3_rsp_rset
Handle POP3 Server reply to RSET command

nx_pop3_rsp_stat
Handle POP3 Server reply to STAT command

nx_pop3_rsp_user
Handle POP3 Server reply to USER command

nx_pop3_session_delete
Delete a POP3 Client session instance

nx_pop3_session_initialize
Initialize a POP3 Client session instance

nx_pop3_session_reinitialize
ReInitialize a POP3 Client session instance for another mail session.

nx_pop3_session_run
Run the POP3 protocol state machine to transmit mail to a POP3 Server and maintain Client state

nx_pop3_utility_print_client_mailitem
Display downloaded mail item message content

nx_pop3_utility_print_client_reserves
Display available packet and memory pool reserves

nx_pop3_client_connect

Connect to Client POP3 Server

Prototype

```
UINT    nx_pop3_client_connect(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service establishes a TCP connection with the Client POP3 server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Client successfully connected to Server
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Connect to the POP3 server. */
status = _nx_pop3_client_connect(session_ptr);

/* If connection was successfully established, status = NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_session_initialize, nx_pop3_session_reinitialize, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_client_delete, nx_pop3_session_delete

nx_pop3_client_create

Create a POP3 Client instance

Prototype

```
UINT    nx_pop3_client_create(NX_POP3_CLIENT *client_ptr,
                  CHAR *client_name, CHAR *client_password,
                  CHAR *shared_secret,
                  UINT APOP_authentication, NX_IP *ip_ptr,
                  NX_PACKET_POOL *packet_pool_ptr,
                  TX_BYTE_POOL *bytepool_ptr,
                  TX_MUTEX *bytepool_mutex_ptr,
                  ULONG bytepool_mutex_timeout,
                  TX_BLOCK_POOL *blockpool_ptr,
                  TX_MUTEX *blockpool_mutex_ptr,
                  ULONG blockpool_mutex_timeout,
                  ULONG reply_timeout, ULONG window_size,
                  UINT (*nx_pop3_client_mail_spooler)
                        (NX_POP3_CLIENT_MAIL *mail_ptr))
```

Description

This service creates an instance of the POP3 Client.

Input Parameters

client_ptr	Pointer to Client to create
client_name	Pointer to Client name
client_password	Pointer to Client password
client_shared_secret	Pointer to Client shared secret
APOP_authentication	Enable APOP authentication
ip_ptr	Pointer to Client IP instance
packet_pool_ptr	Pointer to Client packet pool
bytepool_ptr	Pointer to Client bytepool
bytepool_mutex_ptr	Pointer to Client bytepool mutex
bytepool_mutex_timeout	Timeout for obtaining bytepool mutex
blockpool_ptr	Pointer to Client blockpool
blockpool_mutex_ptr	Pointer to Client blockpool mutex
blockpool_mutex_timeout	Timeout for obtaining blockpool mutex
reply_timeout	Timeout for receiving server reply
window_size	TCP window size for receiving data
nx_pop3_client_mail_spooler	Pointer to callback mail spooler function

Return Values

NX_SUCCESS	(0x00)	Client successfully created
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Application code

Example

```
/* Create the POP3 Client requiring Client block and byte pool memory. Note
   that the Client uses its password for its APOP shared secret. */
status =  nx_pop3_client_create(&demo_client, LOCALHOST, LOCALHOST_PASSWORD,
                                LOCALHOST_PASSWORD,
                                NX_FALSE /*NX_POP3_CLIENT_ENABLE_APOP */,
                                &client_ip, &client_packet_pool,
                                &client_byte_pool, &client_byte_pool_mutex, 100,
                                &client_block_pool, &client_block_pool_mutex, 100,
                                NX_POP3_CLIENT_REPLY_TIMEOUT,
                                NX_POP3_CLIENT_WINDOW_SIZE, client_mail_spooler);

/* If the Client was successfully created, status = NX_SUCCESS. */
```

See Also

nx_pop3_client_delete, nx_pop3_session_initialize, nx_pop3_session_reinitialize, nx_pop3_session_delete

nx_pop3_client_delete

Delete a POP3 Client instance

Prototype

```
UINT    nx_pop3_client_delete(NX_POP3_CLIENT *client_ptr)
```

Description

This service deletes a previously created POP3 Client.

Input Parameters

client_ptr	Pointer to Client to delete

Return Values

NX_SUCCESS	(0x00)	Client successfully deleted
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Application code

Example

```
/* Delete the POP3 Client. */
status =  nx_pop3_client_delete (&demo_client);
/* If the Client was successfully deleted, status = NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_session_initialize, nx_pop3_session_reinitialize, nx_pop3_session_delete

nx_pop3_cmd_dele

Sends a DELE command to POP3 server

Prototype

```
UINT    nx_pop3_client_dele(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted DELE command to the Client's POP3 server with the mail item to delete specified by the session maildrop_index field.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the DELE command to the POP3 Server. */
status =  nx_pop3_cmd_dele (session_ptr);

/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_greeting

Set up the connection to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_greeting(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service creates the TCP socket for the Client to communicate, binds it to the POP3 port, and attempts to establish a connection with (“greet”) the POP3 Server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Client successfully greets the Server
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Attempt to connect with the POP3 server. */
status = nx_pop3_cmd_greeting(session_ptr);
/* If the greeting was successful, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_list

Sends a LIST command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_list(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted LIST command to the Client's POP3 server with the mail item to list specified by the session maildrop_index field (or if zero all mail items listed).

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the LIST command to the POP3 server. */
status = nx_pop3_cmd_list(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_noop

Sends a NOOP command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_noop(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted NOOP command to the Client's POP3 server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the NOOP command to the POP3 server. */
status = nx_pop3_cmd_noop(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_pass

Sends a PASS command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_pass(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted PASS command to the Client's POP3 server with the Client password as the only argument.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the PASS command to the POP3 server. */
status = nx_pop3_cmd_pass(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_quit

Sends a QUIT command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_quit(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted QUIT command to the Client's POP3 server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the QUIT command to the POP3 server. */
status = nx_pop3_cmd_quit(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_retr

Sends a RETR command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_retr(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted RETR command to the Client's POP3 server with the mail item to retrieve based on the session maildrop_index field.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the RETR command to the POP3 server. */
status = nx_pop3_cmd_retr(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_rset

Sends a RSET command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_rset(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted RSET command to the Client's POP3 server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the RSET command to the POP3 server. */
status = nx_pop3_cmd_rset(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_stat

Sends a STAT command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_stat(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted STAT command to the Client's POP3 server.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the STAT command to the POP3 server. */
status = nx_pop3_cmd_stat(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_cmd_user

Sends a USER command to the POP3 Server

Prototype

```
UINT    nx_pop3_cmd_user(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service sends a properly formatted USER command to the Client's POP3 server with the Client name as the only argument. However if the Client is configured to use APOP authentication, this service will send the APOP command with the Client name and MD5 digest of Client password and Server ID instead.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Command successfully sent
NX_POP3_ILLEGAL_CLIENT_COMMAND	(0xB2)	Illegal command in session state
NX_POP3_APOP_FAILED_MD5_DIGEST	(0xB5)	Server rejects APOP authentication
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Send the USER command to the POP3 server. */
status = nx_pop3_cmd_user(session_ptr);
/* If the command was successfully sent, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit,

nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset,
nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_rsp_user

nx_pop3_mail_add

Adds a mail item to the Client session mail list

Prototype

```
UINT    nx_pop3_mail_add(NX_POP3_CLIENT_SESSION *session_ptr,
                         NX_POP3_CLIENT_MAIL *mail_ptr)
```

Description

This service adds a newly created mail item to the list of mail in the current Client session. This mail item will be used to store mail retrieved from the POP3 server. This service is only used for Clients configured to use Client byte and block pool memory allocation.

Input Parameters

session_ptr	Pointer to Client session
mail_ptr	Pointer to mail item to add

Return Values

NX_SUCCESS	(0x00)	Mail item added to session list
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Add the specified mail item to the session mail list. */
status = nx_pop3_cmd_user(session_ptr, mail_ptr);
/* If the mail was successfully added, status = NX_SUCCESS. */
```

See Also

nx_pop3_mail_create, nx_pop3_mail_delete, nx_pop3_rsp_retr, nx_pop3_message_segment_add, nx_pop3_session_delete, nx_pop3_mail_spool

nx_pop3_mail_create

Create a Client mail instance

Prototype

```
UINT    nx_pop3_mail_create(NX_POP3_CLIENT_SESSION *session_ptr,
                            NX_POP3_CLIENT_MAIL **session_mail_ptr)
```

Description

This service creates a mail item for storing and spooling mail retrieved from the POP3 server. This service is only used for Clients configured to use Client byte and block pool memory allocation.

Input Parameters

session_ptr	Pointer to Client session
session_mail_ptr	Pointer in memory allocated for mail item to create

Return Values

NX_SUCCESS	**(**0x00)	Mail successfully created
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Create a mail item using Client byte pool memory. */
status = nx_pop3_mail_create(session_ptr, mail_ptr);
/* If the mail was successfully created, status = NX_SUCCESS. */
```

See Also

nx_pop3_mail_add, nx_pop3_mail_delete, nx_pop3_rsp_retr, nx_pop3_message_segment_add, nx_pop3_session_delete, nx_pop3_mail_spool

nx_pop3_mail_delete

Delete a Client mail instance

Prototype

```
UINT    nx_pop3_mail_delete(NX_POP3_CLIENT_SESSION *session_ptr,
                            NX_POP3_CLIENT_MAIL *mail_ptr)
```

Description

This service deletes a mail item previously created in a Client POP3 session. For Clients configured to use Client byte and block pool memory allocation, it releases memory allocated for creating the mail item. For all clients, the mail instance is then cleared in memory.

Input Parameters

session_ptr	Pointer to Client session
mail_ptr	Pointer to mail item to delete

Return Values

NX_SUCCESS	(0x00)	Mail item successfully deleted
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Delete the specified mail item. */
status = nx_pop3_mail_delete(session_ptr, mail_ptr);
/* If the mail was successfully deleted, status = NX_SUCCESS. */
```

See Also

nx_pop3_mail_add, nx_pop3_mail_create, nx_pop3_rsp_retr, nx_pop3_message_segment_add, nx_pop3_session_delete, nx_pop3_mail_spool

nx_pop3_mail_spool

Spool a mail item to hard disk

Prototype

```
UINT    nx_pop3_mail_spool(NX_POP3_CLIENT_SESSION *session_ptr,
                           NX_POP3_CLIENT_MAIL *mail_ptr)
```

Description

This service spools a mail item which contains a mail data retrieved from the Client POP3 server to hard disk so memory used to store the mail data can be used for subsequent mail item retrieval in the current Client session. This is a user defined callback service.

Input Parameters

session_ptr	Pointer to Client session
mail_ptr	Pointer to mail item to spool

Return Values

NX_SUCCESS	(0x00)	Mail item successfully spooled
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Spool the specified mail item. */
status = nx_pop3_mail_spool(session_ptr, mail_ptr);
/* If the mail was successfully spooled, status = NX_SUCCESS. */
```

See Also

nx_pop3_mail_add, nx_pop3_mail_create, nx_pop3_mail_delete, nx_pop3_rsp_retr, nx_pop3_message_segment_add, nx_pop3_session_delete

nx_pop3_rsp_dele

Handle Server reply to Client DELE command

Prototype

```
UINT    nx_pop3_rsp_dele(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent DELE command. It verifies the server accepted the command and saves Server data (number of maildrop items and total maildrop message data) to the session.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client DELE command. */
status = nx_pop3_rsp_dele(session_ptr);
/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_delenx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_greeting

Handle Server reply to Client greeting

Prototype

```
UINT    nx_pop3_rsp_greeting(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to the Client's attempt to connect to the POP3 server. It verifies the server accepted the command and sets the Client session state to the Authorization state.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_APOP_NO_SERVER_PID	(0xB4)	Server greeting has no process ID
NX_POP3_SERVER_REJECTS_COMMAND	(0xBA)	Server rejects Client command
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client greeting. */
status = nx_pop3_rsp_greeting(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_list

Handle Server reply to Client LIST command

Prototype

```
UINT    nx_pop3_rsp_list(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent LIST command. It verifies the server accepted the command and saves Server data (number of maildrop items and total maildrop message data if no specific mail item specified in the LIST command or current mail item size if one was) to the session.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_BAD_SERVER_LIST_REPLY	(0xB6)	Improperly formatted LIST reply
NX_POP3_CANNOT_PARSE_REPLY	(0xB9)	Unable to parse Server reply
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client LIST command. */
status = nx_pop3_rsp_list(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_noop

Handle Server reply to Client NOOP command

Prototype

```
UINT    nx_pop3_rsp_noop (NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent NOOP command. It verifies the server accepted the command.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client NOOP command. */
status = nx_pop3_rsp_noop(session_ptr);
/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_pass

Handle Server reply to Client PASS command

Prototype

```
UINT    nx_pop3_rsp_pass(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent PASS command. It verifies the server accepted the command and the Client password and sets the Client session state to the Transaction state.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_SERVER_REJECTS_COMMAND	(0xBA)	Server rejects Client command
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client PASS command. */
status = nx_pop3_rsp_pass(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_quit

Handle Server reply to Client QUIT command

Prototype

```
UINT    nx_pop3_rsp_quit(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent QUIT command. It verifies the server accepted the command and enables the session status to end with successful completion status.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	**(**0x00)	Reply successfully processed
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client QUIT command. */
status = nx_pop3_rsp_quit(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_retr

Handle Server reply to Client RETR command

Prototype

```
UINT    nx_pop3_rsp_retr(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent RETR command. It verifies the server accepted the command. If so it will spool the mail to hard disk and delete the mail instance used to store the mail item specified in the RETR command.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_FAILED_PACKET_EXTRACTION	(0xBB)	Packet data extraction failed
NX_POP3_CANNOT_PARSE_REPLY	(0xB9)	Unable to parse Server reply
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client RETR command. */
status = nx_pop3_rsp_retr(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_rset

Handle Server reply to Client RSET command

Prototype

```
UINT    nx_pop3_rsp_rset(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent RSET command. It verifies the server accepted the command.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_SERVER_REJECTS_COMMAND	(0xBA)	Server rejects Client command
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client RSET command. */
status = nx_pop3_rsp_rset(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_stat

Handle Server reply to Client STAT command

Prototype

```
UINT    nx_pop3_rsp_stat(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent STAT command. It verifies the server accepted the command and saves the data in the Server reply (number of items in Client maildrop and total amount of mail data) to the Client session.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	**(**0x00)	Reply successfully processed
NX_POP3_CANNOT_PARSE_REPLY	(0xB9)	Unable to parse Server reply
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client STAT command. */
status = nx_pop3_rsp_stat(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_cmd_user, nx_pop3_rsp_user

nx_pop3_rsp_user

Handle Server reply to Client USER command

Prototype

```
UINT    nx_pop3_rsp_user(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service handles the Client POP3 Server reply to a previously sent USER command. It verifies the server accepted the command. If so, then if the actual USER command was sent, it sets the session to send the PASS command next. If the APOP command was sent, it sets the session state to the Transaction state.

Input Parameters

session_ptr	Pointer to Client session

Return Values

NX_SUCCESS	(0x00)	Reply successfully processed
NX_POP3_CANNOT_PARSE_REPLY	(0xB9)	Unable to parse Server reply
NX_POP3_SERVER_REJECTS_COMMAND	(0xBA)	Server rejects Client command
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Process the POP3 server reply to Client USER command. */
status = nx_pop3_rsp_user(session_ptr);

/* If the server reply was successfully processed, status = NX_SUCCESS. */
```

See Also

nx_pop3_cmd_dele, nx_pop3_rsp_dele, nx_pop3_cmd_greeting, nx_pop3_rsp_greeting, nx_pop3_cmd_list, nx_pop3_rsp_list, nx_pop3_cmd_noop, nx_pop3_rsp_noop, nx_pop3_cmd_pass, nx_pop3_rsp_pass, nx_pop3_cmd_quit, nx_pop3_rsp_quit, nx_pop3_cmd_retr, nx_pop3_rsp_retr, nx_pop3_cmd_rset, nx_pop3_rsp_rset, nx_pop3_cmd_stat, nx_pop3_rsp_stat, nx_pop3_cmd_user

nx_pop3_session_delete

Delete a POP3 Client session Instance

Prototype

```
UINT    nx_pop3_session_delete(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service deletes a POP3 session including any session mail downloaded during the session. It also disconnects from the POP3 server, unbinds and deletes the session socket, and terminates and deletes the session thread.

Input Parameters

session_ptr	Pointer to POP3 session to delete

Return Values

NX_SUCCESS	**(0x00)**	Client session successfully deleted
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Delete the POP3 session instance. */
 status =  nx_pop3_session_delete(session_ptr);

/* If a POP3 session instance was successfully deleted, status is NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_client_delete, nx_pop3_session_initialize, nx_pop3_session_reinitialize, nx_pop3_mail_create, nx_pop3_mail_delete, nx_pop3_session_run

nx_pop3_session_initialize

Initialize a POP3 Client session Instance

Prototype

```
UINT    nx_pop3_session_initialize(
                            NX_POP3_CLIENT_SESSION *session_ptr,
                            ULONG session_id,
                             NX_POP3_CLIENT *client_ptr,
                            UINT delete_downloaded_mail,
                            ULONG ip_addr, USHORT port)
```

Description

This service initializes the Client POP3 session and assigns the session parameters for connecting to the POP3 server and setting various session options.

Input Parameters

session_ptr	Pointer to POP3 session to initialize
session_ID	session ID
client_ptr	Client associated with the session
delete_downloaded_mail	Enable deleting mail after download
ip_addr	Client IP instance for connecting to server
port	POP3 Server port to connect to

Return Values

NX_SUCCESS	(0x00)	Client session successfully initialized
NX_POP3_PARAM_ERROR	(0xB1)	Invalid non pointer input
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads, Application code

Example

```
/* Initialize the POP3 session instance. */
 status =  nx_pop3_session_initialize(session_ptr);

/* If a POP3 session instance was successfully initialized, status is NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_client_delete, nx_pop3_session_delete, nx_pop3_session_initialize, nx_pop3_mail_create, nx_pop3_mail_delete, nx_pop3_session_run

nx_pop3_session_reinitialize

Reinitialize a POP3 Client session Instance

Prototype

```
UINT    nx_pop3_session_reinitialize(NX_POP3_CLIENT_SESSION
                    *session_ptr, UINT session_availability)
```

Description

This service reinitializes a POP3 session for re-use. It deletes any session mail downloaded during the previous session. It also disconnects from the POP3 server, unbinds and deletes the session socket so that the session can reconnect to the POP3 server for another POP3 session.

Input Parameters

session_ptr	Pointer to POP3 session to delete
session_availability	Indicate if session is immediately available

Return Values

NX_SUCCESS	(0x00)	Client session successfully deleted
NX_POP3_PARAM_ERROR	(0xB1)	Invalid non pointer input
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Reinitialize the POP3 session instance. */
 status =  nx_pop3_session_reinitialize(session_ptr);

/* If a POP3 session instance was successfully reinitialized, status is NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_client_delete, nx_pop3_session_initialize, nx_pop3_session_delete, nx_pop3_mail_create, nx_pop3_mail_delete, nx_pop3_session_run

nx_pop3_session_run

Runs a POP3 Client Session

Prototype

```
UINT    nx_pop3_session_run(NX_POP3_CLIENT_SESSION *session_ptr)
```

Description

This service runs a POP3 session with the Client POP3 server. It sends a series of commands to the POP3 server and handles each server reply with the matching command handler. On completion of the session, it returns status to the caller indicating successful status or abnormal termination. It is up to the caller to reinitialize the session (delete session mail, if any, and disconnect and delete the session socket).

Input Parameters

session_ptr	Pointer to POP3 session to delete

Return Values

NX_SUCCESS	(0x00)	Client session successfully deleted
NX_POP3_PARAM_ERROR	(0xB1)	Invalid non pointer input
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Threads

Example

```
/* Initiate and conduct a POP3 session. */
 status =  nx_pop3_session_run (session_ptr);

/* If a POP3 session instance was successfully reinitialized, status is NX_SUCCESS. */
```

See Also

nx_pop3_client_create, nx_pop3_client_delete, nx_pop3_session_initialize, nx_pop3_session_reinitialize, nx_pop3_session_delete, nx_pop3_mail_create, nx_pop3_mail_delete

nx_pop3_utility_print_client_mailitem

Display the Client mail item message

Prototype

```
VOID    nx_pop3_utility_print_client_mailitem(NX_POP3_CLIENT_MAIL *mail_ptr)
```

Description

This service prints the mail message that is part of the POP3 Client mail item pointed to by the specified mail pointer.

Input Parameters

mail_ptr	Pointer to POP3 mail item

Return Values

None

Allowed From

Threads, Application code

Example

```
/* Display Client mail item. */
nx_pop3_utility_print_client_mailitem(mail_ptr);

/* If a valid mail_ptr was supplied, mail message is displayed. */
```

See Also

nx_pop3_utility_print_client_reserves

nx_pop3_utility_print_client_reserves

Display the Client memory pool and packet pool reserves

Prototype

```
VOID    nx_pop3_utility_print_client_reserves(NX_POP3_CLIENT *client_ptr)
```

Description

This service prints the available memory in the Client byte and block pools, and available packets in the Client packet pool. If the Client is not configured to use dynamic memory allocation (has no byte or block pool) , only packet pool availability is displayed.

Input Parameters

client_ptr	Pointer to POP3 Client

Return Values

None

Allowed From

Threads, Application code

Example

```
/* Display Client reserves. */
nx_pop3_utility_print_client_reserves (client_ptr);

/* If a valid client_ptr was supplied, output is displayed. */
```

See Also

nx_pop3_utility_print_client_mailitem

Chapter 4 Description of POP3 Server Services

Services for Server Session and Mail Setup

nx_pop3_server_create
Create a POP3 Server Instance

nx_pop3_server_delete
Delete a POP3 Server instance

nx_pop3_server_session_create
Create a POP3 Server session instance

nx_pop3_server_session_delete
Delete a POP3 Server session instance

Services for the POP3 Protocol and Server State Machine

nx_pop3_server_session_reinitialize
Reinitialize a POP3 session for Client connection

nx_pop3_server_session_run
Service Client commands in a POP3 session

nx_pop3_server_session_start
Start POP3 Server for accepting Client connections

nx_pop3_server_session_stop
Halt POP3 Server (do not accept Client connections)

nx_pop3_reply_to_apop
Reply to POP3 Client APOP command

nx_pop3_reply_to_dele
Reply to POP3 Client DELE command

nx_pop3_reply_to_greeting
Reply to greeting from POP3 Client

nx_pop3_reply_to_list
Reply to POP3 Client LIST command

nx_pop3_reply_to_noop
Reply to POP3 Client NOOP command

nx_pop3_reply_to_quit
Reply to POP3 Client QUIT command

nx_pop3_reply_to_pass
Reply to POP3 Client PASS command

nx_pop3_reply_to_retr
Reply to POP3 Client RETR command

nx_pop3_reply_to_rset
Reply to POP3 Client RSET command

nx_pop3_reply_to_stat
Reply to POP3 Client STAT command

nx_pop3_reply_to_user
Reply to POP3 Client USER command

nx_pop3_utility_ print_server_reserves
Display available Server memory and Server packet pool packets

Services for the POP3 Server Callback Functions

nx_pop3_server_get_time
Create local time on POP3 Server string

nx_pop3_server_get_PID
Create process ID on POP3 Server string

nx_pop3_server_get_auth_check
Handle a null authentication check callback

nx_pop3_server_create

Create a POP3 Server Instance

Prototype

```
UINT  nx_pop3_server_create(
                NX_POP3_SERVER *server_ptr,
                NX_IP *ip_ptr, VOID *stack_ptr,
                ULONG stack_size, UINT server_priority,
                UINT server_preempt_threshold, UINT server_time_slice,
                UINT auto_start, NX_PACKET_POOL *packet_pool_ptr,
                TX_BYTE_POOL *bytepool_ptr,
                TX_MUTEX bytepool_mutex_ptr,
                UINT bytepool_mutex_timeout,
                UINT (*get_clock_time)(CHAR *clock_time),
                UINT (*get_process_ID)(CHAR *process_ID),
                UINT *authentication_check)(
                        NX_POP3_SERVER_SESSION *session_ptr,
                        CHAR *username, CHAR *password, UINT *result),
                UINT (*get_client_maildrop)(
                        NX_POP3_SERVER_SESSION *session_ptr,
                        CHAR *username),
                UINT (*get_client_mailitem_data)(
                        NX_POP3_SERVER_SESSION *session_ptr,
                        UINT maildrop_index, ULONG *mailitem_bytes),
                UINT (*create_client_maildrop_list)(
                        NX_POP3_SERVER *server_ptr),
                UINT (*get_mail_message_buffer)(
                        NX_POP3_SERVER_SESSION *session_ptr,
                        UINT mailitem_index, CHAR **buffer_ptr,
                        UINT *bytes_extracted, UINT *bytes_remaining),
                UINT (*delete_mail_on_file)(
                        NX_POP3_SERVER_SESSION *session_ptr)))
```

Description

This service creates a POP3 Server instance on the specified IP instance.

Input Parameters

server_ptr	Pointer to POP3 Server control block
ip_ptr	Pointer to Server IP instance
stack_ptr	Pointer to stack location of Server thread
stack_size	Size of stack memory for the Server
server_priority	Server thread priority
server_preempt_threshold	Server thread preemption threshold
server_time_slice	Time slice allocated by the scheduler for Server thread execution
auto_start	Server thread start option
packet_pool_ptr	Pointer to Server packet pool
bytepool_ptr	Pointer to Server byte pool
bytepool_mutex_ptr	Pointer to Server byte pool mutex
bytepool_mutex_timeout	Time to wait to obtain Server byte pool mutex
get_clock_time	Pointer to the get Server clock time service

get_process_ID	Pointer to the get Server process ID service
authentication_check	Pointer to the Server authentication check
get_client_maildrop	Pointer to the get Client maildrop service
create_client_maildrop_list	Pointer to the get list of Client maildrop mail items service
get_mail_message_buffer	Pointer to the get part or all of Client mail item message data service
delete_mail_on_file	Pointer to the delete Client mail item from Server hard drive/cache service

Return Values

NX_SUCCESS	(0x00)	POP3 Server successfully created
NX_PTR_ERROR	(0x16)	Invalid pointer input parameter
NX_POP3_PARAM_ERROR	(0xB1)	Invalid non pointer input parameter

Allowed From

Application Code

Example

```
/* Create the POP3 Server instance. */

   status = nx_pop3_server_create(&demo_server, &clientserver_ip,
                  free_memory_pointer, NX_POP3_SERVER_THREAD_STACK_SIZE,
                  NX_POP3_SERVER_THREAD_PRIORITY,
                  NX_POP3_SERVER_PREEMPTION_THRESHOLD,
                  NX_POP3_SERVER_THREAD_TIME_SLICE, TX_DONT_START,
                  &server_packet_pool, &server_byte_pool, &server_byte_pool_mutex,
                  NX_POP3_SERVER_BYTE_POOL_MUTEX_WAIT, server_get_clock_time,
                  server_get_process_ID, server_authentication_check,
                  server_get_client_maildrop, server_get_client_mailitem_data,
                  server_create_client_maildrop_list, server_get_mail_message_buffer,
                  server_delete_mail_on_file))

/* If a Server was successfully created, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_delete, nx_pop3_server_session_create, nx_pop3_server_session_reinitialize, nx_pop3_server_session_delete, nx_pop3_server_session_run, nx_pop3_server_start, nx_pop3_server_stop

nx_pop3_server_delete

Delete a POP3 Server Instance

Prototype

```
UINT nx_pop3_server_delete(NX_POP3_SERVER *server_ptr);
```

Description

This service deletes a previously created POP3 Server instance.

Input Parameters

client_ptr Pointer to POP3 Server instance.

Return Values

NX_SUCCESS	(0x00)	Server successfully deleted.
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service

Allowed From

Threads

Example

```
/* Delete the POP3 Server instance “my_server.” */
status = nx_pop3_server_delete(&my_server);

/* If the POP3 Server instance was successfully deleted, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_create, nx_pop3_server_session_create, nx_pop3_server_session_reinitialize, nx_pop3_server_session_delete, nx_pop3_server_session_run, nx_pop3_server_start, nx_pop3_server_stop

nx_pop3_server_session_create

Create a POP3 Server Session Instance

Prototype

```
UINT  _nx_pop3_server_session_create(NX_POP3_SERVER *server_ptr,
                    NX_POP3_SERVER_SESSION *session_ptr,
                    UINT session_id, VOID *session_stack_ptr,
                    ULONG session_stack_size,
                    UINT session_priority,
                    UINT session_preempt_threshold,
                    ULONG session_time_slice,
                    UINT session_auto_start)
```

Description

This service creates a POP3 Session instance to accept a POP3 Client connection and conduct a POP3 mail session.

Input Parameters

server_ptr	Pointer to the POP3 Server session
session_ptr	Pointer to POP3 Server session to create
session_id	Session ID
session_stack_ptr	Location of Server session on the stack
session_stack_size	Size of Server session stack memory
session_priority	Server session thread priority
session_preempt_theshold	Server session thread preemption threshold
session_time_slice	Time slice the scheduler allots for Server session thread execution
session_auto_start	Server session thread start option

Return Values

NX_SUCCESS	(0x00)	Server session successfully created
NX_PTR_ERROR	(0x16)	Invalid input pointer parameter

Allowed From

Application code

Example

```
/* Create the POP3 Server Session instance. */
    status = nx_pop3_server_session_create(&demo_server,
                        &(demo_server.nx_pop3_server_session_list[i]),
                        i+1, free_memory_pointer,
                        NX_POP3_SERVER_SESSION_THREAD_STACK_SIZE,
                        NX_POP3_SERVER_SESSION_THREAD_PRIORITY,
```

```
                                NX_POP3_SERVER_SESSION_THREAD_PRIORITY,
                                NX_POP3_SERVER_SESSION_THREAD_TIME_SLICE, TX_DONT_START);

    /* If a POP3 Server session instance was successfully created, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_create, nx_pop3_server_delete, nx_pop3_server_session_delete, nx_pop3_server_session_reinitialize, nx_pop3_server_session_run, nx_pop3_server_start, nx_pop3_server_stop

nx_pop3_server_session_delete

Delete a POP3 Server session Instance

Prototype

```
UINT    nx_pop3_server_session_delete(NX_POP3_SERVER_SESSION
                                      *session_ptr)
```

Description

This service deletes a POP3 Server session instance and all mail associated with the session. It releases all memory dynamically allocated for the session instance back to the Server's memory pools.

Input Parameters

session_ptr	Pointer to POP3 Server session to delete

Return Values

NX_SUCCESS	(0x00)	Server session successfully deleted
NX_CALLER_ERROR	(0x11)	Invalid caller of this service
NX_PTR_ERROR	(0x16)	Invalid pointer parameter

Allowed From

Threads

Example

```
/* Delete the POP3 Server session instance. */
 status =  nx_pop3_server_session_delete(session_ptr);

/* If a Server session instance was successfully deleted, status is NX_SUCCESS. */
```

See Also

nx_pop3_server_create, nx_pop3_server_delete, nx_pop3_server_session_create, nx_pop3_server_session_reinitialize, nx_pop3_server_session_run, nx_pop3_server_start, nx_pop3_server_stop

nx_pop3_server_session_reinitialize

Reinitialize a POP3 session for another Client connection

Prototype

```
UINT    nx_pop3_server_session_reinitialize(NX_POP3_SERVER_SESSION
                                            *session_ptr)
```

Description

This service resets the session attributes and POP3 protocol state to initial values for accepting another Client connection.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Session successfully completed
NX_PTR_ERROR	(0x16)	Invalid pointer parameter
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reinitialize a POP3 Server session with a POP3 Client. */
status = nx_pop3_server_session_reinitialize(session_ptr);

/* If a POP3 Session completes normally, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_create, nx_pop3_server_delete
nx_pop3_server_session_create, nx_pop3_server_session_delete,
nx_pop3_server_session_run, nx_pop3_server_start, nx_pop3_server_stop

nx_pop3_server_session_run

Run a POP3 session to receive mail

Prototype

```
UINT    nx_pop3_server_session_run(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service is the NetX POP3 Server protocol engine. It accepts a POP3 Client request and conducts a POP3 session with that Client. A successful session status is one that completes when the Client issues the QUIT command, regardless whether any actual mail is downloaded to the Client.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Session successfully completed
NX_PTR_ERROR	(0x16)	Invalid pointer parameter
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Execute a POP3 Server session with a POP3 Client. */
status = nx_pop3_server_session_run(NX_POP3_SERVER_SESSION* session_ptr);

/* If a POP3 Session completes normally, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_create, nx_pop3_server_delete
nx_pop3_server_session_create, nx_pop3_server_session_delete,
nx_pop3_server_session_reinitialize, nx_pop3_server_start,
nx_pop3_server_stop

nx_pop3_server_reply_to_apop

Reply to POP3 Client APOP command

Prototype

```
UINT    nx_pop3_server_reply_to_apop(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client APOP command, and determines if the MD5 digest sent with the APOP command contains both the string the Server sent to the Client in the greeting message and the shared secret between Client and Server. If so, it accepts the command and advances the POP3 session state to the transaction state.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client APOP command. */
status = nx_pop3_server_reply_to_apop (session_ptr);

/* If the Server handles the APOP command with no errors regardless if Client is
     authenticated, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_dele

Reply to POP3 Client DELE command

Prototype

```
UINT    nx_pop3_server_reply_to_dele(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client DELE command, and verifies the mail item number received in the command exists in the Client maildrop. If so, it marks that mail item for deletion from the maildrop (Server hard disk).

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client DELE command. */
status = nx_pop3_server_reply_to_dele(session_ptr);

/* If the Server handles the DELE command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_greeting

Reply to POP3 Client greeting

Prototype

```
UINT    nx_pop3_server_reply_to_greeting(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client greeting (connection request), and replies with a Server reply code indicating if it accepts the connection, as well as additional text to greet the Client.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client greeting. */
status = nx_pop3_server_reply_to_greeting(session_ptr);

/* If the session able to send a reply message to Client, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_list

Reply to POP3 Client LIST command

Prototype

```
UINT    nx_pop3_server_reply_to_list(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client LIST command, and verifies the mail item number received in the command exists in the Client maildrop. If so, it displays that mail item number and the number of bytes in the message. If no mail item is contained in the command, it displays a listing for all mail items in the maildrop.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client LIST command. */
status = nx_pop3_server_reply_to_list(session_ptr);

/* If the Server handles the LIST command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_noop

Reply to POP3 Client NOOP command

Prototype

```
UINT    nx_pop3_server_reply_to_noop(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client NOOP command, and replies with the +OK acknowledgment.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client NOOP command. */
status = nx_pop3_server_reply_to_noop(session_ptr);

/* If the Server handles the NOOP command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_pass

Reply to POP3 Client PASS command

Prototype

```
UINT    nx_pop3_server_reply_to_pass(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client PASS command, and performs an authentication check on the Client username and password. If the Client is authenticated, the POP3 session is advanced to the transaction state.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client PASS command. */
status = nx_pop3_server_reply_to_pass(session_ptr);

/* If the Server handles the PASS command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_quit

Reply to POP3 Client QUIT command

Prototype

```
UINT    nx_pop3_server_reply_to_quit(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client QUIT command, and advances the POP3 session to the update state.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client QUIT command. */
status = nx_pop3_server_reply_to_quit(session_ptr);

/* If the Server handles the QUIT command with no errors, status = NX_SUCCESS. */
```

See Also

nx__pop3_x_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_retr

Reply to POP3 Client RETR command

Prototype

```
UINT    nx_pop3_server_reply_to_retr(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client RETR command, and if a valid mail item index is included with the command, will download the requested mail item to the Client.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client RETR command. */
status = nx_pop3_server_reply_to_retr(session_ptr);

/* If the Server handles the RETR command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_rset

Reply to POP3 Client RSET command

Prototype

```
UINT    nx_pop3_server_reply_to_rset(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client RSET command, and if the POP3 session reaches the update state, clears all mail items marked for deletion.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client RSET command. */
status = nx_pop3_server_reply_to_rset(session_ptr);

/* If the Server handles the RSET command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_stat, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_stat

Reply to POP3 Client STAT command

Prototype

```
UINT    nx_pop3_server_reply_to_stat(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client STAT command, and advances the POP3 session to the update state.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client STAT command. */
status = nx_pop3_server_reply_to_stat(session_ptr);

/* If the Server handles the STAT command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_user, nx_pop3_server_session_run

nx_pop3_server_reply_to_user

Reply to POP3 Client USER command

Prototype

```
UINT    nx_pop3_server_reply_to_user(NX_POP3_SERVER_SESSION *session_ptr)
```

Description

This service receives the POP3 Client USER command, and if a valid username is included with the command, accepts the command and waits for the password in the next command.

Input Parameters

session_ptr	Pointer to POP3 Server session instance

Return Values

NX_SUCCESS	(0x00)	Server reply successfully sent
NX_PTR_ERROR	(0x16)	Invalid POP3 Session pointer.
NX_CALLER_ERROR	(0x11)	Invalid caller of this service.

Allowed From

Threads

Example

```
/* Reply to Client USER command. */
status = nx_pop3_server_reply_to_user (session_ptr);

/* If the Server handles the USER command with no errors, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_stat, nx_pop3_server_session_run

nx_pop3_utility_print_server_reserves

Displays the POP3 Server memory and packet pool

Prototype

```
UINT nx_pop3_utility_print_server_reserves(
                                    NX_POP3_SERVER *server_ptr)
```

Description

This service prints the available bytes in server byte pool and available packets in the Server packet pool.

Input Parameters

server_ptr	Pointer to POP3 Server

Return Values

NX_SUCCESS	(0x00)	Server data accessed and displayed successfully
NX_PTR_ERROR	(0x16)	Invalid pointer parameter
NX_CALLER_ERROR	(0x11)	Invalid caller of this service

Allowed From

Threads

Example

```
/* Print Server memory and packet pool reserves. */
Status = nx_pop3_utility_print_server_reserves(server_ptr);

/* If server data was successfully displayed, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_reply_to_greeting, nx_pop3_server_reply_to_apop, nx_pop3_server_reply_to_dele, nx_pop3_server_reply_to_list, nx_pop3_server_reply_to_noop, nx_pop3_server_reply_to_pass, nx_pop3_server_reply_to_retr, nx_pop3_server_reply_to_rset, nx_pop3_server_reply_to_quit, nx_pop3_server_reply_to_stat, nx_pop3_server_session_run

nx_pop3_server_get_time

Get local time on POP3 Server

Prototype

```
UINT nx_pop3_server_get_time(NX_POP3_SERVER_SESSION *session_ptr,
                             CHAR *clock_time)
```

Description

This service directs the service request to the application defined callback *nx_pop3_server_get_clock_time*, or if no callback was supplied, uses the NX_POP3_SERVER_DEFAULT_TIME to create a string containing a static 'local time.' It is up to the caller to allocate storage for the *clock_time* string.

Input Parameters

server_ptr	Pointer to POP3 Server
clock_time	Pointer to buffer for clock time string

Return Values

NX_SUCCESS	(0x00)	Server local time successfully retrieved
NX_PTR_ERROR	(0x16)	Invalid pointer parameter
NX_CALLER_ERROR	(0x11)	Invalid caller of this service

Allowed From

Threads

Example

```
/* Get local clock time on POP3 Server. */
Status = nx_pop3_server_get_time(session_ptr, clock_time);

/* If clock time successfully retrieved, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_get_PID, nx_pop3_server_get_auth

nx_pop3_server_get_PID

Get a process ID on POP3 Server

Prototype

```
UINT nx_pop3_server_get_PID(NX_POP3_SERVER_SESSION *session_ptr,
                            CHAR *process_ID)
```

Description

This service directs the service request to the application defined callback *nx_pop3_server_get_process_ID*, or if no callback was supplied, uses the NX_POP3_SERVER_DEFAULT_PROCESS_ID to create a string containing a static 'process ID.' It is up to the caller to allocate storage for the *process_ID* string.

Input Parameters

server_ptr	Pointer to POP3 Server
process_ID	Pointer to buffer for process ID string

Return Values

NX_SUCCESS	(0x00)	Server PID successfully retrieved
NX_PTR_ERROR	(0x16)	Invalid pointer parameter
NX_CALLER_ERROR	(0x11)	Invalid caller of this service

Allowed From

Threads

Example

```
/* Get session process ID on POP3 Server. */
Status = nx_pop3_server_get_PID(session_ptr, process_ID);

/* If the session process ID successfully retrieved, status = NX_SUCCESS. */
```

See Also

nx_pop3_server_get_time, nx_pop3_server_get_auth